Full Materi

by Ilyas Learning Progaming

Submission date: 21-Mar-2019 02:17AM (UTC-0700)
Submission ID: 1097136558
File name: Linear_Programming_dengan_R.pdf (5.62M)
Word count: 36345
Character count: 214090

Ilyas Masudin
Muhammad Faisal Ibrahim
Gilang Yandeza

LINEAR PROGRAMMING DENGAN R

(Aplikasi untuk Teknik Industri)

UMM PRESS

Ilyas Masudin
Muhammad Faisal Ibrahim
Gilang Yandeza

LINEAR PROGRAMMING DENGAN R
(Aplikasi untuk Teknik Industri)

Penerbit Universitas Muhammadiyah Malang

LINEAR PROGRAMMING DENGAN R

(Aplikasi untuk Teknik Industri)

Penerbit Universitas Muhammadiyah Malang
Jl. Raya Tlogomas No. 246 Malang 65144
Telepon 0877 0166 6388, (0341) 464318 Psw. 140
Fax. (0341) 460435
E-mail: ummpress@gmail.com
http://ummpress.umm.ac.id
Anggota APPTI (Asosiasi Penerbit Perguruan Tinggi Indonesia)
Anggota IKAPI (Ikatan Penerbit Indonesia)

Cetakan Pertama, Juli 2018
ISBN: 978-979-796-335-4

xii; 172 5hlm: 16 x 23 cm

Setting & Layout: AH. Riyantono

Sanksi Pelanggaran Pasal 113
Undang-Undang Nomor 28 Tahun 2014
tentang Hak Cipta

(1) Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).

(2) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah).

(3) Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf a, huruf b, huruf e, dan/atau huruf g untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan/atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah).

(4) Setiap Orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan/atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah).

PRAKATA

Seiring berkembangnya bahasa pemrograman yang sangat cepat, menuntut kecepatan perkembangan keilmuwan lain yang menyertainya. Salah satu disiplin keilmuan teknik industri adalah optimasi. Disiplin ilmu ini erat kaitannya dengan pendekatan matematis yang mendekati sebuah permasalahan keteknikindustrian dari perspektif model matematis. Untuk mendapatkan hasil optimal dengan model matematis ini, salah satu metode yang paling populer dan banyak digunakan hampir di semua disiplin keilmuan di berbagai bidang adalah program linier (*linier programming*).

Buku ini akan mencoba mengkombinasikan penyelesaian problem keteknikindustrian (*problem solving*) yang umum dijumpai di *linier programming* dengan pendekatan matematis dan heuristik dengan penyelesaian menggunakan bahasa pemrograman R. Dalam buku ini, *linier programming* akan dibahas ke dalam 7 pembahasan misalnya 2 bab pertama membahas pengantar bahasa R dan pengertian dari optimasi. Pada bab 3 dan 4 membahas tentang *linier programming* dan penyelesaian grafis, simpleks dan problem transportasi yang disertai penyelesaian menggunakan R. Kemudian dilanjutkan Bab 5 yang mendiskusikan penyelesaian problem programa linier dengan pendekatan penugasan, yang dilanjutkan dengan bab 5 yang mendiskusikan penyelesaian *linier programming* dan bahasa R untuk problem *integer programming*. Buku ini ditutup dengan pembahasan tentang model linier program yang lain yakni *transhipment problem*.

Dalam proses penyelesaian buku ini, penulis banyak mendapatkan bantuan dari berbagai pihak sehingga ucapan terima kasih yang sebesar-besarnya kami ucapkan kepada Direktur DPPM-Universitas Muhammadiyah Malang (UMM), para pimpinan universitas dan fakultas teknik – UMM, para kolega di jurusan teknik industri - UMM, dan para akademisi dan mahasiswa yang tidak terlibat langsung dalam penulisan namun punya sumbangsih besar dalam menginspirasi buku ini. Terakhir, saya mengucapkan terima kasih yang tidak terhingga kepada istri dan anak saya selalu men-*support* dalam buku ini.

Besar harapan kami bahwa semoga buku ini akan memperkaya *body of knowledge* disiplin ilmu optimasi dan disiplin ilmu lainnya yang relevan serta akan memberi manfaat sebesar-besarnya kepada akademisi dan praktisi yang bekerja dengan displin ilmu terkait.

Malang, 5 Juni 2018

Ilyas Masudin

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENGANTAR BAHASA PEMROGRAMAN R

Sejarah R
Kelebihan Menggunakan R
Batasan R
Tentang Free Software
Pemasangan (Instalasi) R
Bagaimana R Bekerja ?
Menjalankan R
Operasi Dasar dan Obyek pada R
Data Vektor dan Matriks pada R
Interface dan Pemrograman Grafik R
Manual Penggunaan R

PENGANTAR BAHASA PEMROGRAMAN R

"Dengan kemajuan teknologi informasi saat ini, sangat tidak relevan jika masih mengolah data secara manual. Bahkan, saat ini komputer bukan hanya sebagai alat pengolah data, namun dapat sebagai 'tools' untuk memecahkan masalah, dan menyajikannya dalam bentuk informasi".

Gambar 1. 1 Logo R (Source : R-Project)

Bahasa R (berikutnya dalam buku ini hanya akan disebut dengan "R" saja), yaitu fasilitas software yang dapat digunakan dalam manipulasi data, simulasi, kalkulasi dan juga peragaan grafik (R Project n.d.). R dapat dipakai untuk menyelesaikan dan melakukan analisa data, R juga dapat digunakan untuk pengelolaan data berbentuk array dan matr. Selain itu, R juga menyediakan alat bantu statistik untuk pemodelan linear dan non linear, uji statistik klasik, klasifikasi, *clustering*, analisis *time series*, dan bahkan mampu untuk menampilkan grafik untuk peragaan dari data yang telah diolah. R, adalah penerapan GNU (General Public Licence) dari bahasa S. Bahkan saat ini, R telah menjadi *platform* serbaguna dan kuat untuk melakukan analisis statistik (Handoyo, Prasojo, & Naba, 2017).

Sejarah R

Bahasa R dibuat berdasarkan basis bahasa S yang dikembangkan di Bell Laboratories pada tahun 1980-an. Karena itu R memiliki sintak yang hampir sama dengan baha S. Ross Ihaka dan Robert Gentelman berhasil membuat versi pertama R pada tahun 1992 di Universitas Auckland, New aland. Singkatan R berasal dari kedua nama pembuatnya tersebut. Selanjutnya R dikembangkan oleh tim inti (R-core team) yang terdiri dari 17 ahli statistik, komputer dan pemrograman dari berbagai institusi (R Project n.d.), yang mempunyai tujuan untuk membuat sebuah software yang handal dengan biaya murah. Selain itu, berbagai ahli lain dari berbagai negara berbeda di luar n inti tersebut juga berkontribusi dengan melakukan pengembangan kode, melaporkan bug, dan membuat dokumentasi manual penggunaan R. Di awal pengembangannya, R hanya dapat diterapkan pada machintosh dengan bahasa LISP. Namun sekarang, R telah menjadi multiplatform, dengan begitu dapat digunakan di berbagai sistem operasi lainnya (linux, Mac OS) dan Windows.

Kelebihan Menggunakan R

R memiliki kelebihan-kelebihan yang memudahkan penggunaannya, adapun alasan menggunakan R menurut Ihaka & Gentleman (1996), yaitu :

1. Se Guna

R merupakan salah satu bahasa pemrograman yang memiliki banyak package, sehingga para penggunanya tidak terbatas pada paket standarnya saja. Bahasa R telah berorientasi pada obyek dan memiliki

library. Adapun *library* tersebut telah dikembangkan oleh kontributor bahasa R. Banyaknya *library* tersebut menjadi sangat bermanfaat bagi pengguna. *Library* yang telah diunduh dapat digunakan dengan bebas, pengguna mungkin saja akan mengurangi atau menambahkan bahasa yang telah ada dalam *library* tersebut sesuai kebutuhan pengguna. Tidak hanya itu kontributor juga telah berhasil membuat R memiliki interface pemrograman lainnya, contohnya phyton, C, dan java. Pengguna R memiliki kesempatan berkreasi dan menyebarkan *library* yang telah mereka buat. Tentunya tidak semua analisis yang akan pengguna lakukan telah ada *library*nya. *Package* dalam *library* belum tersedia mungkin saja karena belum ada pengguna lain sebagai kontributor yang pernah mengerjakan analisa sejenis dengan R.

2. Sangat Interaktif

Era sekarang ini membutuhkan pengoperasian yang interaktif. Sebagai contoh, saat penguna ingin melakukan analisa data yang dinamis. Tentunya software yang digunakan harus mampu terkoneksi ke sebuah database. R telah mampu terkoneksi ke database server, olap, spreadsheet, dan lain sebagainya. Dengan begitu, akan sangat memungkinakan melakukan analisa dengan R secara *realtime*.

3. Berbasis S

Karena R kompatibel dengan S-Plus sehingga hampir semua kode program yang dibuat oleh R memungkinkan untuk dijalankan di S-plus. Walaupun begitu memang kode-kode tertentu tidak kompatibel. Umumnya fungsi-fungsi yang tidak kompatibel tersebut adalah fungsi tambahan, yaitu yang dibuat oleh kontributor proyek R.

4. Populer

Walaupun menurut peneliti di bidang statistika SAS adalah software statistika yang paling populer, namun R atau S tetap menjadi bahasa yang paling populer digunakan oleh peneliti di bidang statistika. Kebenaran fakta tersebut telah terkonfirmasi dari beberapa penelitian yang menggunakan bahasa R atau S. Tidak hanya itu, bidang keuangan juga meningkatkan popularitas R karena dapat digunakan pada bidang keuangan.

Batasan R

Terlepas dari kemudahan dalam penggunaan R, tidak ada bahasa pemrograman dan sistem analisis statistik yang

sempurna. Itu juga yang dirasakan pengguna terhadap R. Walaupun bahasa R dianggap sederhana dan umum, namun menurut pengguna bahasa R juga tidak mudah untuk dipelajari. Sulitnya mempelajari bahasa R umum terjadi apabila pengguna tidak memiliki basic bahasa pemrograman sama sekali. Sehingga diperlukan pemahaman dasar-dasar pemrograman sebelum dapat menggunakan R dengan lebih mudah. Dukungan secara komersial pun tidak tersedia, namun hal ini dapat diatasi dengan *mailing-list* internasional. Dan juga R memiliki antarmuka untuk grafik yang terbatas, dimana dalam hal ini S-Plus lebih unggul (Ihaka & Gentleman 1996).

Tentang Free Software

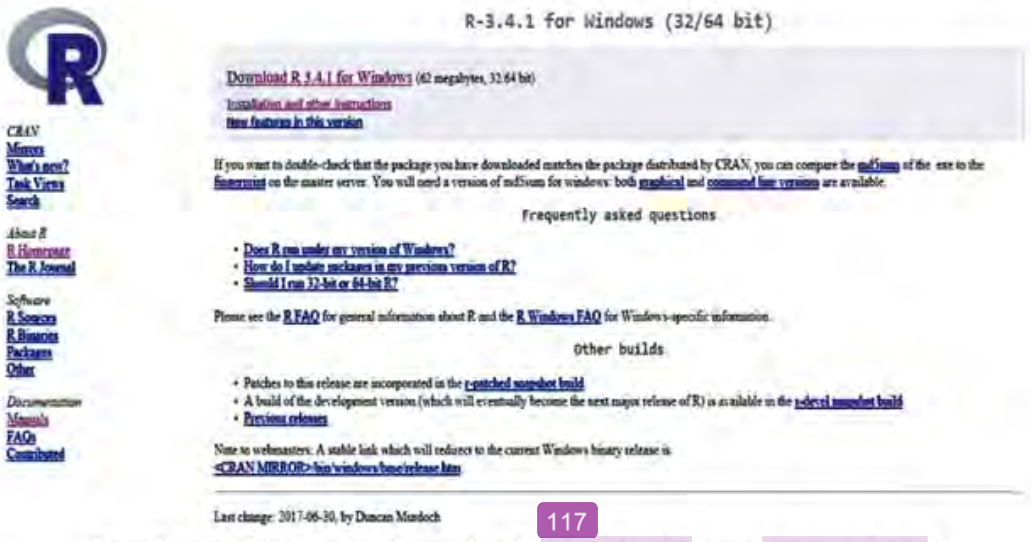
Seperti yang telah diketahui bahwa manfaat dari penggunaan R yang meliputi berbagai *packages* adalah bersifat *open* dan *free*. Hak cipta dari *source code* utama R dipegang oleh R Foundation dan dipublikasikan dibawah GNU General Public License Version 2.0.

Hal-hal yang dapat kita peroleh dari status *free software* adalah sebagai berikut.

1. Kebebasan untuk menjalankan program untuk kegunaan apapun *(freedom 0)*.
2. Kebebasan untuk belajar bagaimana program bekerja, dan menyesuaikan kebutuhan *(freedom 1)*
3. Kebebasan untuk mendistribusikan ulang salinan dari program untuk membantu yang lainnya *(freedom 2)*
4. Kebebasan untuk memperbaiki program dan merilis perbaikan tersebut ke publik *(freedom 3)*.

Pemasangan (Instalasi) R

R adalah versi *open-source* dari bahasa pemrograman S, dimana S adalah bahasa pemrograman yang dikembangkan oleh John Chambers dan lainnya pada tahun 1976. R dapat diperoleh secara gratis pada situs http://cran.r-project.org/ . Saat ini (Juli 2017), versi R yang sudah dirilis adalah versi R 3.4.1 tersedia untuk platform Windows, Linux dan Mac OS. Secara umum proses untuk pemasangan sangat mudah untuk dilakukan (Lihat: https://cran.r-project.org/doc/manuals/r-release/R-admin.html).



Gambar 1. 2 Halaman Download (http://www.cran.r-project.org)

Setelah memilih *Download R 3.4.1. for Windows* (yang berarti pada sistem operasi Windows, R rilis terbaru adalah versi 3.4.1.), simpan file di suatu direktori tertentu dan secara otomatis file akan didownload. Selanjutnya, klik dua kali pada file yang didownload untuk memulai pemasangan R. Lanjutkan seperti instalasi software pada Windows umumnya.

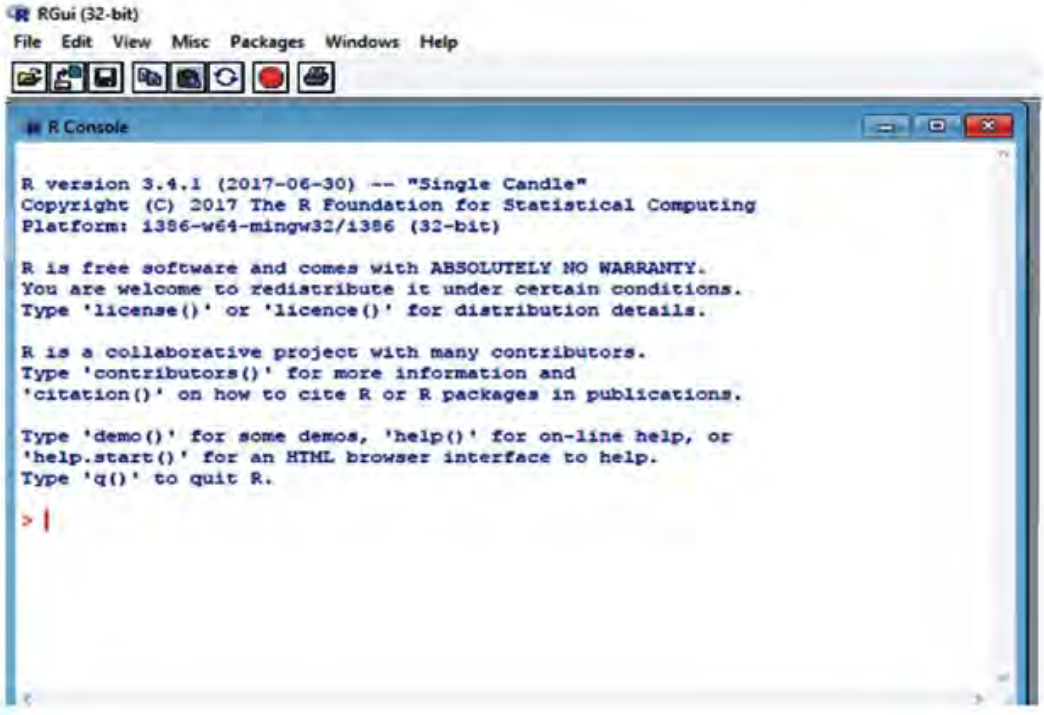
Bagaimana R Bekerja ?

Karena bahasa pemrograman R berorientasi objek, sehingga seluruh peubah, data, fungsi, hasil dan seluruh proses yang dilakukan akan disimpan berupa objek yang memiliki nama dalam memori aktif komputer. Operator-operator dan fungsi yang dipilih akan memberikan perlakukan terhadap objek. Adapun contoh operator-operator yang digunakan adalah aritmatik, logikal, dan pembanding. Perintah yang dijalankan oleh R akan melakukan perlakuan tertentu pada objek-objek yang ada pada memori aktif komputer tampa menggunakan file temporer (*temporary file*). Proses membaca dan menulis file hanya digunakan untuk input dan ouput data dan hasil (grafik). Hasil eksekusi yang dilakukan pengguna dengan perintah-perintah tertentu akan langsung ditampilkan, setelah itu akan disimpan sebagai objek. Hasil yang telah menjadi objek tersebut dapat digunakan dan dianalisa sebagai data pada umumnya untuk perintah selanjutnya. File data tersebut juga telah dapat dibaca langsung melalui server (jika data berada di internet) atau pada lokal disk.

Direktori bernama R_HOME/library (R_HOME adalah direktori dimana R terpasang) disiapkan pada local disk untuk menyimpan fungsi-fungsi yang telah disediakan untuk pengguna. Fungsi-fungsi *packages* yang telah ada maupun yang akan ditambahkan akan terseimpan dalam direktori tersebut. Adapun package bernama base adalah package basic R. Package tersebut berisi fungsi-fungsi dasar yang digunakan dalam bahasa R. Adapun contoh fungsi dasar tersebut antara lain adalah fungsi membaca, manipulasi data, grafik, dan statistik. Package lainnya akan berada pada direktori berbeda dengan nama sesuai dengan package tersebut. Sebagai contoh file package base berada pada R_HOME/library/base/R/base.

Menjalankan R

Proses instalasi secara umum tidak berlangsung lama. Ketika program R dijalankan, maka tampilan awal dari R biasa disebut R *Console*, dimana tulisan berwarna biru menunjukkan versi dari R, dan beberapa perintah dasar seperti *license(), contributor(), demo()*, dan sebagainya.



Gambar 1. 3 Tampilan R Console

R adalah bahasa pemrograman yang berbasis pada *command line* (yakni berupa baris berisi perintah) yang ditulis pada konsol. Tanda > dapat diartikan prompt. Tanda tersebut akan selalu muncul tanpa perlu diketik manual. Prompt berguna sebagai penunjuk lokasi perintah-perintah yang akan diketikkan. Tidak jarang perintah yang ditulis cukup panjang sehingga tidak mencukupi dalam 1 baris. Pengguna dapat melanjutkan perintah di baris selanjutnya dengan menekan enter, tanda + akan muncul sebagai prompt, arti prompt (+) adalah baris tersebut adalah lanjutan dari baris sebelumnya.

Contohnya, ketika ingin melakukan penjumlahan dari 1+25 atau mengetahui nilai dari Π misalnya, cukup mengetik di konsol sebagai berikut.

```
> 1+25
> pi
```

Maka akan menghasilkan seperti pada gambar berikut.

Gambar 1. 4 Contoh Perintah Pada Console R

Operasi Dasar dan Obyek pada R

Pada R, seperti yang telah dicontohkan pada sub bab sebelumnya, dapat berfungsi sebagai kalkulator aritmetika

sederhana. Sebagai contoh.

Gambar 1. 5 Contoh Aritmetika Pada R

Kode yang dituliskan dibelakan tanda "#" akan diartikan sebagai komentar. Sebagai contoh pada gambar 1.5 dapat dilihat semua kode yang ditulis dibelakang tanda # tidak diproses dan diartikan sebagai komentar. Selanjutnya ketika ingin memberi atau menyimpan nilai pada sebuah variabel, dapat dilakukan dengan notasi "<- atau ->", untuk versi 1.4 ke atas dapat dengan operator "=", namun notasi "<- atau ->" lebih jelas yang berarti memasukkan suatu nilai ke dalam variabel.
Misalnya,

```
> n<-17
> n
[1] 17
atau
> 17->N
> N
[1] 17
> n=15
# nilai objek n yang lama (17) dihapus & diganti dengan 15
> n
[1] 15
```

Case sensitive R perlu diperhatikan karena R sensitif terhadap sangat sensitif terhadap perbedaanpenulisan sekecil apapun. Sebagai contoh, R sensitif terhadap huruf kecil dan besar sehingga pengguna harus memperhatikan cara penulisan mereka. Sehingga obyek x dan X akan diartikan obyek berbeda. Selain dengan menyimpan nilai secara langsung pada obyek, dapat juga merupakan hasil dari suatu operasi dan/atau fungsi.

```
> n<-7+9
> n
[1] 16
> n<-2+rnorm(1) # fungsi rnorm(1) membangkitkan sebuah perubah acak normal baku
> n
[1] 2.723687
```

Selain itu, kita juga dapat menuliskan ekspresi tanpa menyimpan hasilnya ke objek (memori), tetapi menampilkan secara langsung ke layar.

```
> (1+4)*3
[1] 15
> sqrt(81) # fungsi akar kuadrat
[1] 9
```

Terdapat 4 jenis obyek standar pada R, yaitu vektor, matriks, list dan data frame (Handoyo Samingun & Purwanto Ari 2017) . Obyek-obyek ini dibedakan karena struktur data yang digunakannya. Vektor adalah suatu array satu dimensi, matriks adalah array multi dimensi sedangkan data frame dan list adalah obyek yang dapat memuat beberapa obyek dengan jenis yang berbeda.

Data Vektor dan Matriks pada R

Telah dijabarkan secara detail didalam bab sebelumnya bahwa data vektor adalah suatu array satu dimensi. Dengan menggunakan bahasa R, ada dua cara untuk membuat vektor, yaitu dengan fungsi c() dan seq(). Untuk memasukkan beberapa data numerik dan karakter sembarang ke suatu variabel, misalnya bernama jumlah, dapat digunakan fungsi c() (*concatenate*) sebagai berikut.

```
>jumlah <-c(10,23,6)
> jumlah
[1] 10 23 6
```

Untuk menampilkan data pada indeks ke 3

```
> jumlah [3]
[1] 6
```

Jika ingin menampilkan 2 data pertama variabel jumlah.

```
> jumlah [1:2]
[1] 10 23
```

Untuk menjumlahkan nilai dari variabel jumlah,

```
> sum(jumlah)
[1] 33
```

Untuk mencari rata-rata, median, ragam, dan standar deviasi dapat menggunakan beberapa fungsi berikut.

```
> mean(jumlah)          # rata-rata
[1] 13
> median(jumlah)        # median
[1] 10
> var(jumlah)           # variansi/ragam sampel
[1] 79
> sd(jumlah)            # simpangan baku sampel
[1] 8.888194
```

Untuk menampilkan ringkasan statistik dari suatu variabel, diperoleh dengan fungsi summary()

```
> summary(jumlah)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    6.0     8.0    10.0    13.0    16.5    23.0
```

Selanjutnya, selain untuk membuat vektor dalam bentuk numerik, fungsi c() juga dapat digunakan untuk memasukkan data berupa karakter,

```
>Nama <- c("Satu", "Dua", "Tiga", "Empat")
> Nama
[1] "Satu" "Dua" "Tiga" "Empat"
```

Sedangkan fungsi seq() (sequential) adalah fungsi untuk membuat vektor dengan pola khusus, dalam hal ini dapat mengganti notasi ":" bila data mempunyai pertambahan satu nilai. Misalnya,

```
>jumlah1 <- 1:10
>jumlah2 <- seq(1.5, 2.5, by = 0.5)
>jumlah3 <- seq(1.5, 2.5. length = 10)
```

Pada variabel jumlah1, akan menghasilkan sebuah vektor yang bernilai dari 1,2,...,10. Variabel jumlah2 menghasilkan sebuah vektor yang bernilai 1.5, 2, 2.5 dan variabel jumlah3 beranggotakan 10 elemen yang terletak antara 1.5 dan 2.5 dengan tiap elemen mempunyai skala yang sama.

Jika vektor adalah suatu array satu dimensi, maka matriks adalah data multidimensi (dalam buku ini dibahas dua dimensi saja). Bentuk matriks akan sering dipergunakan dalam operasi-operasi dengan R, contohnya pada penyelesaian persamaan linier. Fungsi yang digunakan untuk membuat matriks adalah matrix(). Argumen yang diperlukan dalam membuat sebuah matriks adalah elemen-elemen dari matriks itu sendiri, sedangkan opsional dapat berupa banyaknya baris dan kolom, misalnya,

```
> matriks <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol=3)
> matriks
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Matriks juga dapat diisi secara baris per baris dengan menggunakan argumen opsional byrow = TRUE. Misalnya,

```
data <- c(9, 4, 6, 7, 2, 1)
> matriks = matrix(data, nrow = 3, byrow=TRUE)
> matriks
```

```
     [,1] [,2]
[1,]    9    4
[2,]    6    7
[3,]    2    1
```

Pengguna juga dapat memanfaatkan fungsi pada R untuk menggabungkan sebuah kolom maupun baris yang akan dibuat menjadi matriks. Fungsi cbind() pada R digunakan untuk menambahkan ke dalam sebuah kolom. Sedangkan fungsi rbind() digunakan untuk menambahkan kedalam sebuah baris. Fungsi cbind() digunakan untuk membuat matriks berdasarkan pada beberapa vektor yang panjangnya sama. Vektor-vektor tersebut digabungkan dan masing-masing menjadi kolom-kolom dari matriks. Fungsi rbind() menggabungkan vektor-vektor penyusun sebagai baris-baris dari matriks, misalnya,

```
> a <- matrix(c(1, 2, 9, 0, 5, 7), 2, 3)
> a
     [,1] [,2] [,3]
[1,]    1    9    5
[2,]    2    0    7
> b = cbind(a,c(5,8)) # menambahkan ke kolom ke 4 dari a
> b
     [,1] [,2] [,3] [,4]
[1,]    1    9    5    5
[2,]    2    0    7    8
> c = rbind(a,c(0,3,5)) #menambahkan ke baris ke 3 dari a
> c
     [,1] [,2] [,3]
[1,]    1    9    5
[2,]    2    0    7
[3,]    0    3    5
```

Selain itu terdapat satu jenis data obyek yang memiliki bentuk sama dengan bentuk data matriks namun dalam setiap kolomnya dapat berupa mode data berbeda. Jenis data obyek tersebut adalah data frame. Adapun

contoh jenis dataframe adalah sebagai berikut :

```
> matriks <- matrix(1:9,3)
> dataframe <- data.frame(nomer=1:4, nama=c("A", "B","C","D"),
nilai = 6:9)
> dataframe
  nomer nama nilai
1     1    A     6
2     2    B     7
3     3    C     8
4     4    D     9
```

Interface dan Pemrograman Grafik R

Jika pengguna ingin melakukan edit data melalui software R tentu saja harus memiliki pengetahuan bahasa pemrograman yang cukup. Namun, untuk lebih mudahnya R telah menyediakan cara edit data lainnya yaitu melalui spreadsheet. Pada umumnya, edit data melalui spreadsheet akan jauh lebih mudah. Contoh penggunaannya adalah sebagai berikut.

```
> data.entry(x)   # muncul spreadsheet untuk mengedit data x
> x<-de(x)        # idem, coba perhatikan per bedaannya dengan
mengetik x
> x<-edit(x)      # muncul notepad editor untuk mengedit x
```

Walaupun perintah tersebut cukup mudah untuk digunakan namun pengguna tidak boleh terlewat mendefinisikan obyek data yang ingin dirubah melalui spreadsheet. Bila tidak, sudah pasti pengguna tidak bisa menggunakan perintah tersebut. Berikut ini adalah contoh pesan kesalahan yang muncul akibat objek yang akan diedit belum pernah didefinisikan sebelumnya.

```
> data.entry(z) # gagal, objek z belum terdefinisi
Error in de(..., Modes = Modes, Names = Names) :
Object "z" not found
> data.entry(z=c(NA)) # muncul spreadsheet
> zz<-numeric() # definisikan varibel/objek zz dengan tipe numerik
> data.entry(zz) # muncul spreadsheet
```

Dari dua cara entri data yang dijelaskan dalam buku ini, akan sangat baik jika divisualisasikan dalam bentuk grafik yang akan memudahkan dalam melakukan analisa. R mampu membedakan berbagai macam data secara otomatis. Sebagai contoh data yang umum digunakan adalah tipe data kategori, numerik diskrit dan kontinu.

Yang disebut data kategori adalah data yang umum tersaji dalam tabel, diagram batang, diagram lingkaran, dan diagram sejenis lainnya. Pada data kategori, fungsi table() digunakan menyajikan data katagorikal dalam bentuk tabel. Misalnya, ada sebuah data yang menunjukkan hasil dari pengelompokan jenis kelamin dari satu departemen laki-laki disimbolkan dengan "L", sedangkan perempuan disimbolkan dengan "P".

```
> hasil <- c('L', 'P', 'L', 'L', 'P', 'L', 'P')
> table(hasil)
hasil
L P
4 3
```

Data kategori secara umum dapat diartikan sebagai data yang diklasifikasikan ke dalam beberapa level atau faktor. Sehingga factor() dapat digunakan dalam mendapatkan level dari data kategori.

```
> factor(hasil)
[1] L P L L P L P
Levels: L P
> facthasil <- factor(hasil)
> facthasil
[1] L P L L P L P
Levels: L P
```

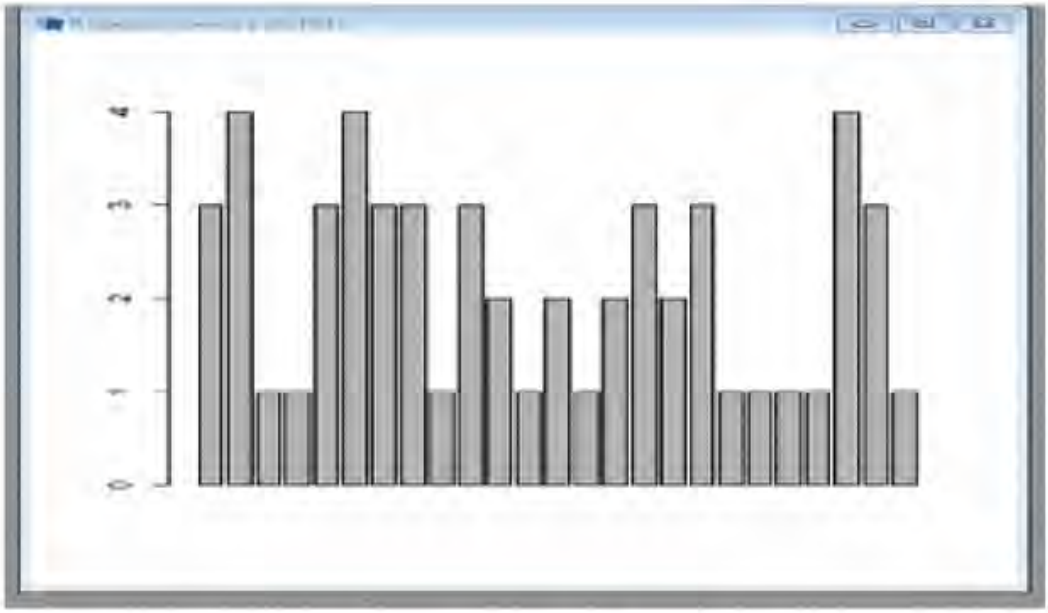
Dengan begitu obyek fact hasil telah mempunyai level, sedangkan obyek hasil tidak mempunyai level. Sebagai contoh, hasil di bawah ini dapat menjelaskan hal tersebut.

```
> levels(facthasil)
[1] "L" "P"
> levels(hasil)
NULL
```

Selanjutnya, data kategori juga dapat ditampilkan dalam bentuk grafik batang. Grafik ini akan menampilkan diagram batang yang menggambarkan pencacahan dalam sebuah tabel yang diaplikasikan fungsi table(). Pada disgram batang tersebut akan tergambar frekuensi maupun proporsi dari data yang dianalisa.

Berikut adalah sebuah data hasil dari jumlah produksi barang harian selama 25 hari, yaitu 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1. Selanjutnya, fungsi scan() digunakan untuk membaca hasil tersebut. Fungsi scan() memudahkan entri data yang bernilai banyak.

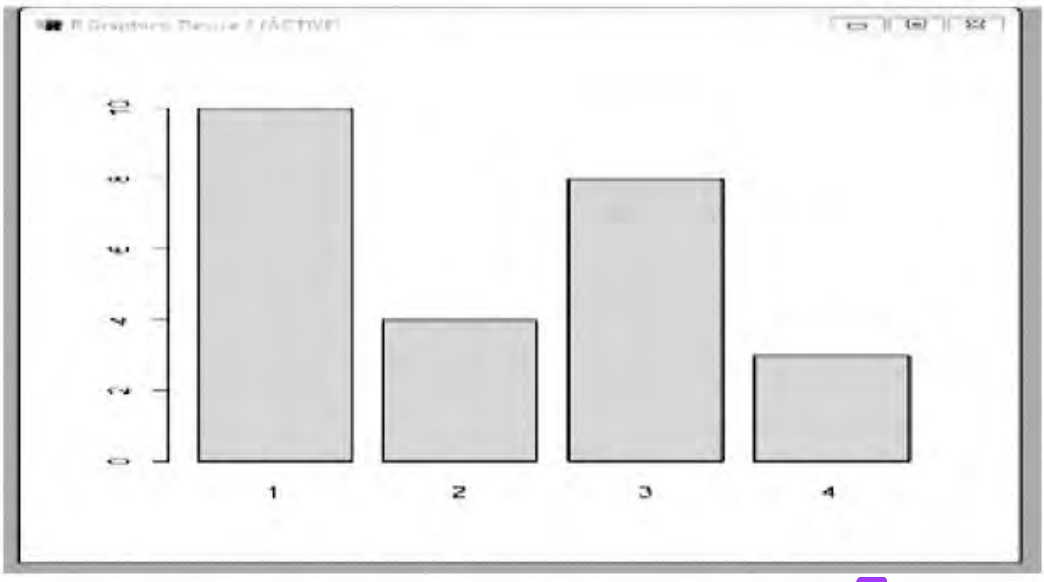
```
> hasil <- scan()
1: 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
26:
Read 25 items
> barplot(hasil)
```



Gambar 1. 6 Hasil Barplot Untuk Data Individu

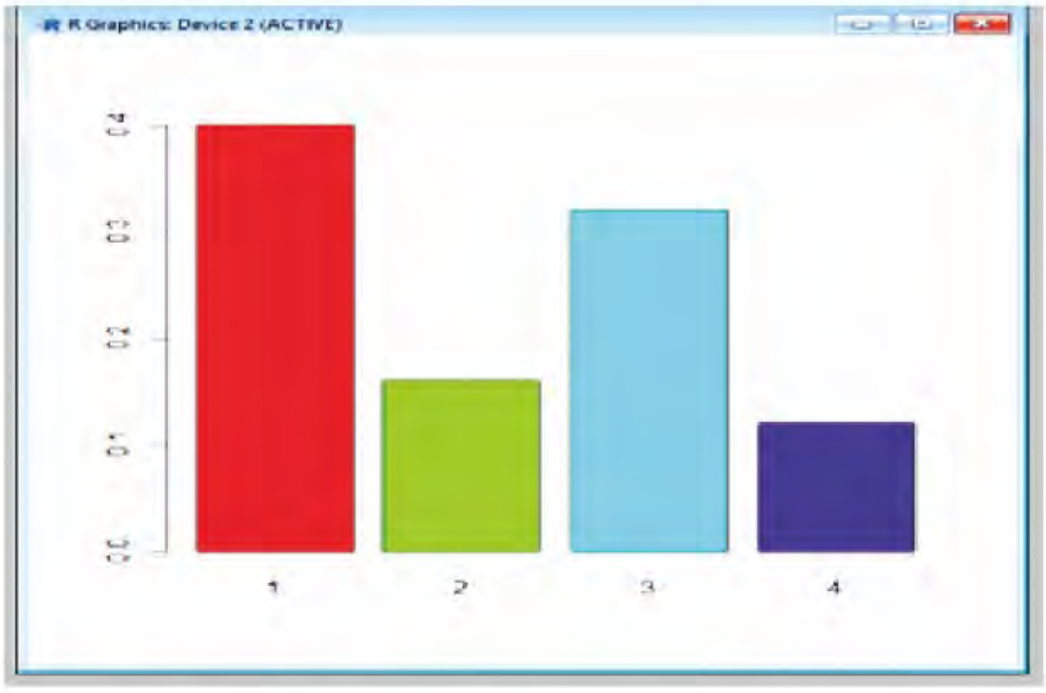
Hasil di atas masih salah, dikarenakan diagram yang ditampilkan adalah hasil dari data individu. Sedangkan untuk data kategori, ditampilkan sebagai berikut.

```
> barplot(table(hasil))
```



Gambar 1. 7 Hasil Barplot Untuk Data Kategori (Berdasarkan Frekuensi)

> barplot(table(hasil)/length(hasil),col=rainbow(4)) #menampilkan proporsi



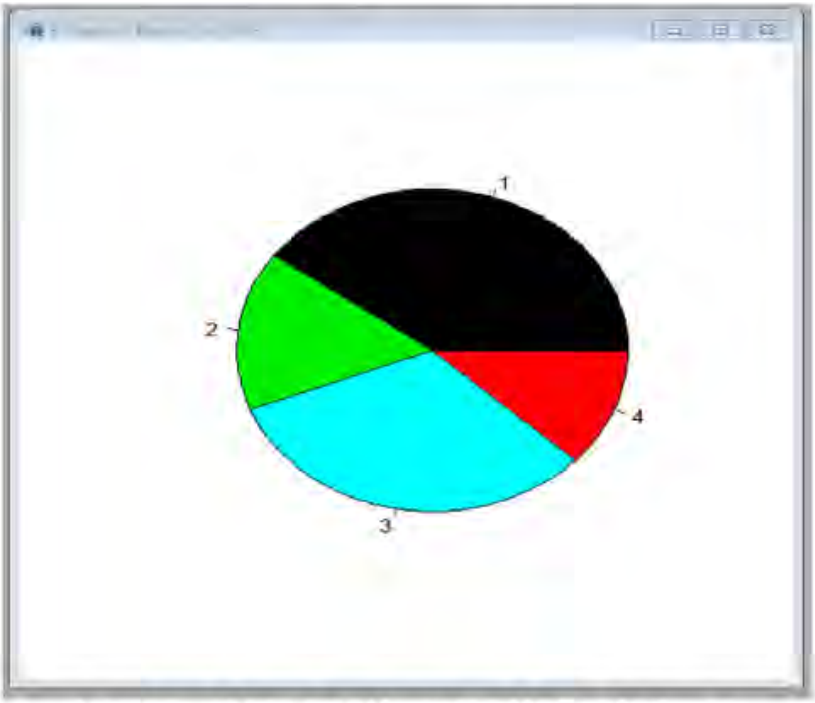
Gambar 1. 8 Hasil Barplot Untuk Data Kategori (Berdasarkan Proporsi)

Pengguna juga dapat membuat variasi warna diagram batang tersebut dengan menggunakan argumen col=rainbow(4). Dengan argumen tersebut maka diagram batang yang telah dibuat akan memiliki variasi warna untuk setiap batangnya. Pengguna juga dapat merubah nilai menjadi proporsional dengan perintah table(hasil)/length(hasil).

```
> table(hasil)/length(hasil)
hasil
1   2   3   4
0.40 0.16 0.32 0.12
```

Pengguna juga dapat membuat diagram lingkaran untuk melakukan analisa data. Sebagai contoh saat pengguna akan menganalisa data yang sama dengan diagram lingkaran, pengguna dapat menggunakan fungsi pie(). Pie chart atau diagram lingkaran adalah grafik yang memiliki bentuk lingkaran atau tersegmentasi, besarnya daerah yang terbagi sesuai dengan frekuensi atau proporsi. Berikut ini adalah contoh penggunaan pie chart.

```
> hasil.frek = table(hasil)
> pie(hasil.frek, col=c('black','green2','cyan','red1')) #Memberi warna
```



Gambar 1. 9 Hasil Pie Chart Untuk Data Kategori

Pada R, jika ingin mengetahui lebih jauh tentang fungsi suatu fungsi, dapat menambahkan notasi "?" di depan nama suatu fungsi, misalnya untuk barplot dan pie, gunakan perintah berikut.

```
> ?barplot
> ?pie
```

Data lainnya yang akan dibahas adalah data numerik. Pengguna R dapat memvisualisasikan data numerik dengan berbagai cara. Sebagai contoh beberapa pilihan cara antara lain adalah ringkasan numerik, pemusatan, dan dispersi. Jika pengguna ingin menggambarkan distribusi data, tentunya sebelumnya perlu terlebih dahulu mencari pusat dan dispersi data. Pencarian pusat dan dispersi data dapat dilakukan dengan menggunakan fungsi mean untuk rata-rata, var untuk variansi, sd untuk standar deviasi, median untuk nilai tengan, atau juga dapat dengan fungsi summary. Fungsi ringkasan dapat memberikan hasil secara keseluruhan. Selain itu fungsi umum yang sering digunakan adalah fungsi data(). Fungsi tersebut dapat digunakan dalam melihat data set apa saja yang tersedia. Contoh penggunaan fungsi data() adalah sebagai berikut.

```
> data() # melihat data terpasang apa saja yang tersedia
> data(iris) # memanggil dataset iris
> iris # melihat isi dari objek dataset iris
```

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
16           5.7         4.4          1.5         0.4     setosa
17           5.4         3.9          1.3         0.4     setosa
18           5.1         3.5          1.4         0.3     setosa
19           5.7         3.8          1.7         0.3     setosa
20           5.1         3.8          1.5         0.3     setosa
21           5.4         3.4          1.7         0.2     setosa
22           5.1         3.7          1.5         0.4     setosa
23           4.6         3.6          1.0         0.2     setosa
24           5.1         3.3          1.7         0.5     setosa
25           4.8         3.4          1.9         0.2     setosa
26           5.0         3.0          1.6         0.2     setosa
27           5.0         3.4          1.6         0.4     setosa
28           5.2         3.5          1.5         0.2     setosa
29           5.2         3.4          1.4         0.2     setosa
30           4.7         3.2          1.6         0.2     setosa
31           4.8         3.1          1.6         0.2     setosa
32           5.4         3.4          1.5         0.4     setosa
33           5.2         4.1          1.5         0.1     setosa
34           5.5         4.2          1.4         0.2     setosa
35           4.9         3.1          1.5         0.2     setosa
36           5.0         3.2          1.2         0.2     setosa
```

Gambar 1. 10 Isi Dari Obyek Dataset Iris

Untuk menampilkan obyek dari Sepal.Length, gunakan perintah iris$Sepal.Length, perhatikan karakter $ sebagai penyambung.

```
> Sepal.Length  # salah, obyek tidak ditemukan
Error: object 'Sepal.Length' not found
> iris$Sepal.Length # obyek Sepal.Length ada di bawah iris
 [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7
5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0 5.5 4.9 4.4 5.1
 [41] 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6
5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7
 [81] 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1
5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0
```

[121] 6.9 5.6 7.7 6.3 6.7 7.2 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8 6.7 6.7 6.3 6.5 6.2 5.9

Ringkasan statistik sederhana dapat ditampilkan menggunakan fungsi summary().

```
> summary(iris)
```

```
Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa     :50
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Pola data yang memiliki jumlah data tidak terlalu banyak dapat dilihat juga dengan menggunakan diagram batang dan daun (stem and leaf atau stemplot). Diagram tersebut dapat menyajikan batang dan daun dari suatu variabel. Batang yang dimaksud dapat berupa puluhan, ratusan atau dalam bentuk pecahan sedangkan daun berarti nilai yang mengikutinya. Sebagai contoh, terdapat sekumpulan data 11, 15, 17 maka dapat dinyatakan dalam bentuk 1|157, dimana angka 1 adalah bilangan puluhan, sedangkan 1,5 dan 7 adalah bilangan satuan. Fungsi stem() akan menampilkan visualisasi berbentuk diagram tersebut. Masih menggunakan dataset iris, berikut ditampilkan diagram stem and leaf dari Sepal.Length

```
> stem(iris$Sepal.Length)
```

```
  The decimal point is 1 digit(s) to the left of the |
  42 | 0
  44 | 0000
  46 | 000000
  48 | 00000000000
  50 | 0000000000000000000
  52 | 00000
  54 | 0000000000000
```

```
56 | 00000000000000
58 | 0000000000
60 | 000000000000
62 | 0000000000000
64 | 000000000000
66 | 0000000000
68 | 0000000
70 | 00
72 | 0000
74 | 0
76 | 00000
78 | 0
```

Keterangan untuk "The decimal point is 1 digit(s) to the left of the |" berarti titik(koma) yang menunjukkan nilai desimal berada pada posisi jika tanda "|" digeser 1 desimal ke kiri. Misalnya angka 42 akan dibaca 4.2 dengan jarak antar batang adalah 0.2 yang merepresentasikan dua titik desimal.

Histogram adalah salah satu jenis grafik yang sering digunakan dalam menggambarkan data numerik. Diagram ini menggambarkan bentuk sebaran data ke dalam distribusi frekuensi Fungsi hist() menghasilkan histogram yang dimaksud.

Tidak hanya fungsi tampilan grafis yang sudah dijelaskan pada buku ini, karana hanya fungsi basic yang dijelaskan dalam buku ini. Masih terdapat bentuk tampilan grafis lainnya yang dapat pengguna pelajari melalui panduan-panduan R.

Manual Penggunaan R

Untuk mengetahui fungsi-fungsi dasar dari R dapat menelusuri di manual penggunaan R yang disediakan oleh R-Project.

http://cran.r-project.org

http://cran.r-project.org/doc/manuals/r-release/R-intro.html

http://cran.r-project.org/doc/manuals/r-release/R-data.html

http://cran.r-project.org/doc/manuals/r-release/R-exts.html

http://cran.r-project.org/doc/manuals/r-release/R-admin.html

http://cran.r-project.org/doc/manuals/r-release/R-ints.html

http://cran.r-project.org/doc/manuals/r-release/R-lang.html

Sistem Fuzzy Terapan Dengan Software R (Handoyo *et al.*, 2017)

The R Project For Statistical Computing (https://www.r-project.org/)

BAB 2
PENGANTAR OPTIMASI

Klasifikasi Model Matematis dalam Optimasi
Model Optimasi Matematis Sederhana: Tanpa Kendala
Optimasi dalam R
Penyelesaian Model Optimasi Sederhana dengan R

PENGANTAR OPTIMASI

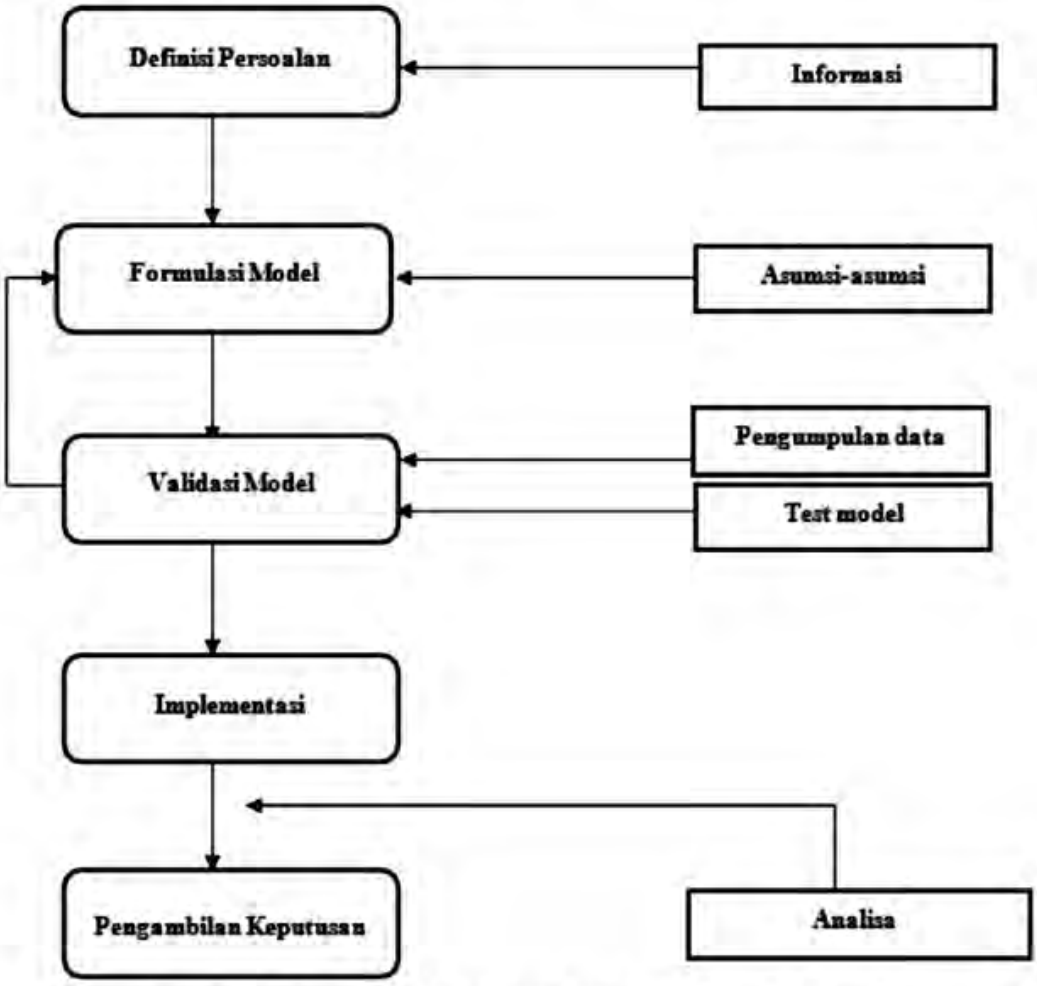
Kata optimasi sangat terkait erat dengan *operations research* (OR) yang menjadi core keilmuan teknik industri, dimana hampir 90% ilmu teknik industri bersumber dan bersandar pada teori optimasi atau OR ini. Berbagai keputusan baik dalam lingkunagn mikro maupun makro dengan keterbatasan tertentu bisa didekati dan diselesaikan dengan pendekatan OR. Itulah kenapa pendekatan OR ini bahkan tidak hanya populer digunakan di jurusan teknik industri namun juga sangat umum digunakan di keilmuwan eksak lain. Bahkan dalam jurnal internasional elit *Management Science*, yang sangat populer dalam ilmu-ilmu ekonomi bahasan tema di dalamnya tidak sedikit dijumpai pendekatan optimasi atau OR ini sebagai pendekatan atau metode yang dpakai dan dikolaborasikan dengan teori-teori ekonomi dan ilmu non eksak lainnya. Hal itu membuktikan bahwa operations research atau optimasi ini bisa dipakai dalam kondisi lingkungan yang bervariasi dan sangat superior menyelesaikan permasalahan secara efektif. Bahkan kolaborasi ilmu ekonomi dan teknik industri (*production research*) melalui pendekatan optimasi atau OR telah mealahirkan sebuah dalam jurnal ekonomi internasional berkualifikasi A seperti *International Journal of Production Economics* (IJPE) yang diterbitkan penerbit terkenal *Elsevier*.

Operations Research (OR) berkenaan dengan pengambilan keputusan secara ilmiah, bagaimana membuat model terbaik dan membeutuhkan sumber daya terbatas. *Operation Research Society of Great Britain* mendefinisikan OR adalah aplikasi metode ilmiah

dalam masalah yang kompleks dan sistem manajemen yang besar, baik yang menyangkut manusia, mesin, bahan (material) dan uang (Wijaya, 2012). Sementara itu *Operations Research Society of America* (ORSA) mendefinisikan operations research adalah berkenaan dengan pengambilan keputusan secara ilmiah, bagaimana membuat model yang terbaik dan membutuhkan alokasi sumberdaya terbatas. Pada awal perang dunia kedua penggunaan analisa kuantitatif dikembangkan oleh Thomas Edison yang menganalisa keefektifan perjalanan zig-zag untuk melindungi kapal dagang dari serangan kapal yang terlibat peperangan. Sejak saat itu selama perang dunia kedua OR benar-benar menjadi metode penyelesaian yang paling efektif. Inggris dikenal sebagai negara pertama yang membentuk secara formal kelompok organisasi OR. Tahun 1939, G.A Robert dan DR. William pertama kali mengembangkan sistem komunikasi untuk angkatan udara Inggris. Setelah Amerika terlibat dalam peperangan, angkatan bersenjata Amerika mulai mengembangkan organisasi OR. Tahun 1942 angkatan udara Amerika membentuk Divisi *Operations Analysis*. Begitu juga angkatan laut Amerika. Sejak saat itu *operations research* secara terus menerus berkembang di bawah nama angkatan bersenjata dan berkembang pada bidang-bidang non militer.

Seiring berkembangnya ilmu teknik industri dan semakin eratnya kolaborasi antara research operations dengan teori ilmu lainnya, maka istilah research operations banyak dikenal menjadi berbagai macam istilah seperti penelitian opersional, manajemen operasional dan optimasi. Di jurusan teknik industri, *operations research* lebih sering dipakai untuk program terkait optimasi sistem industri (OSI) atau biasa disebut optimasi. Optimasi berkenaan dengan pengambilan keputusan secara ilmiah, bagaimana membuat dan melakukan perbaikan yang lebih baik untuk menghasilkan tujuan yang terbaikdengan ketersediaan sumber daya yang terbatas. Dalam aplikasinya, pengambilan keputusan dalam optimasi sistem industri tidak terlepas dari proses pembuatan model baik model grafis maupun model matematis. Proses pemodelan dalam optimasi sistem industri dimulai dari proses pendefinisian permasalahan/problem, yang kemudian diikuti dengan memformulasikan kondisi yang sedang dihadapi (*current situation*) ke dalam bentuk model, yang bisa berupa model matematis, grafis, ataupun model konseptual lainnya. Proses pembuatan model dalam optimasi bisa dilihat dalam Gambar 2.1.

Gambar 2.1 menunjukkan proses pengambilan keputusan dalam teori optimasi atau *operations research*. Model pengambilan keputusan pada tahap awal adalah pengumpulan informasi terkait permasalahan yang akan diselesaikan. Informasi terkait problem yang akan diselesaikan akan membantu pemodel memahami kondisi *current situation* dan permasalahan yang sedang dihadapi. Dengan begitu, proses kedua yakni memformulasikan model dari persoalan yang dihadapi akan bejalan lebih mudah. Dalam proses membuat model baik itu model matematis, model grafis atau bentuk model yang lain, pemodel biasanya memerlukan beberapa asumsi atau keterbatasan model. Hal ini dlakukan untuk mengisolasi kondisi hasil optimasi pada lingkungan terbatas, misalnya batasan kapasitas sumber daya dan material yang dimiliki atau asumsi kepemenuhan permintaan (*demand*).



Gambar 2. 1 Proses Pembuatan Model Dalam Optimasi

Dalam proses memodelkan, asumsi yang dipakai juag harus mempertimbangkan tingkat generalisasi model atau tingkat keumuman model. Karena karakteristik model dikatakan baik menurut Simatupang (2000) antara lain apabila :

1. Mempunyai tingkat generalisasi yang tinggi. Kemudahan sebuah sebuah model dipahami dan diaplikasikan di kasus lain yang relevan menunjukkan bahwa model yang dibuat dikatakan sebagai model yang baik.
2. Mempunyai mekanisme yang transparan. Dalam proses membuat model tentunya pemodel memerlukan informasi selengkap mungkin untuk dipakai membuat model. Kesempurnaan model akan semakin baik jika informasi yang digunakan bisa diterangkan kembali tanpa ada yang disembuyikan.
3. Mempunyai potensi untuk dikembangkan. Sebuah model dikatakan baik adalah jika model tersebut terbuka untuk dikembangkan dalam problem lebih luas. Hal itu menunjukkan bahwa model yang dibuat sanggup menarik peneliti-peneliti dan pemodel lain untuk melakukan penelitian menggunakan ide dasar model semula.
4. Mempunyai kepekaan terhadap perubahan asumsi. Asumsi yang dibuat dalam mebuat model mutlak diperlukan karena lingkungan problem yang dihadapi juga kompleks, namun perlu diingat bahwa asumsi yang dibuat atau di pertimbangkan untuk terlibat dalam sebuah model harus memenuhi unsur kepekaan untuk berubah (*changeable*) menurut situasi.

Pada tahap berikutnya setelah formulasi model adalah validasi model. Sebelum model di*release* atau digunakan atau diaplikasikan ke dalam *real system*, ada satu tahapan vital yang tidak boleh dilewatkan yakni validasi model. Validasi model terkait dengan *model testing* untuk mengetahui seberapa besar kesesuaian model dengan problem sebenarnya. Perlu diketahui bahwa model merupakan representasi dari sebuah sistem nyata yang dikemas atau dibentuk menjadi bentuk/model tertentu sehingga memudahkan pemodel dan oaring lain memahami sistem nyata yang terkadang berbentuk sangat kompleks. Validasi dilakukan dengan tujuan agar risiko kesalahan dan kerugian atau risiko yang yang timbul akibat kesalahan atau ketidakakuratan model yang dibentuk bisa diminimalkan atau dihilangkan saat diaplikasikan di *real system*.

Seperti diketahui model matematis pada pendekatan optimasi menurut Taha (2004), pembentukan model dimulai dengan pembentukan variabel, kendala (*constraint*) dan tujuan (*objective*), sehingga pada pendekatan optimasi untuk model matematik validasi model dilakukan untuk melihat kembali sejauh mana model matematis tersebut dapat diguakan untuk menjawab pertanyaan yang mungkin timbul berkaitan dengan model yang digambarkan (Mananoma & Soetopo, 2009). Sehingga proses validasi pada pendekatan optimasi untuk model matematis bisa dilakukan dengan pengecekan satu persatu variable dan komponen pembentuk model matematis tersebut. Misalnya pada variable keputusan atau *decision variables* yang akan diukur atau dicari optimasinya, atau pengecekan juga juga bisa dilakukan di fungsi pembatas yang membatasi (*constraint*). Unjuk kerja model yang dibuat sebaiknya diuji coba dengan sungguh-sungguh mematuhi kaidah-kaidah proses validasi secara terbuka dan transparan. Pada proses validasi, terdapat proses yang dinamakan "*looping*" dimana pada proses ini, setelah pemodel mendapatkan *feedback* dan saran perbaikan dari proses test model, maka pemodel akan melakukan menyempurnakan modelnya dengan mengakomodasi masukan, perbaikan dan revisi yang diperlukan. Pada pendekatan optimasi untuk model matematis yang cukup kompleks dan memerlukan bantuan software, proses *looping* pada tahap validasi ini biasanya dilakukan software tersebut berupa verifikasi ketepatan dan keakuratan persamaan matematis yang telah dimodelkan.

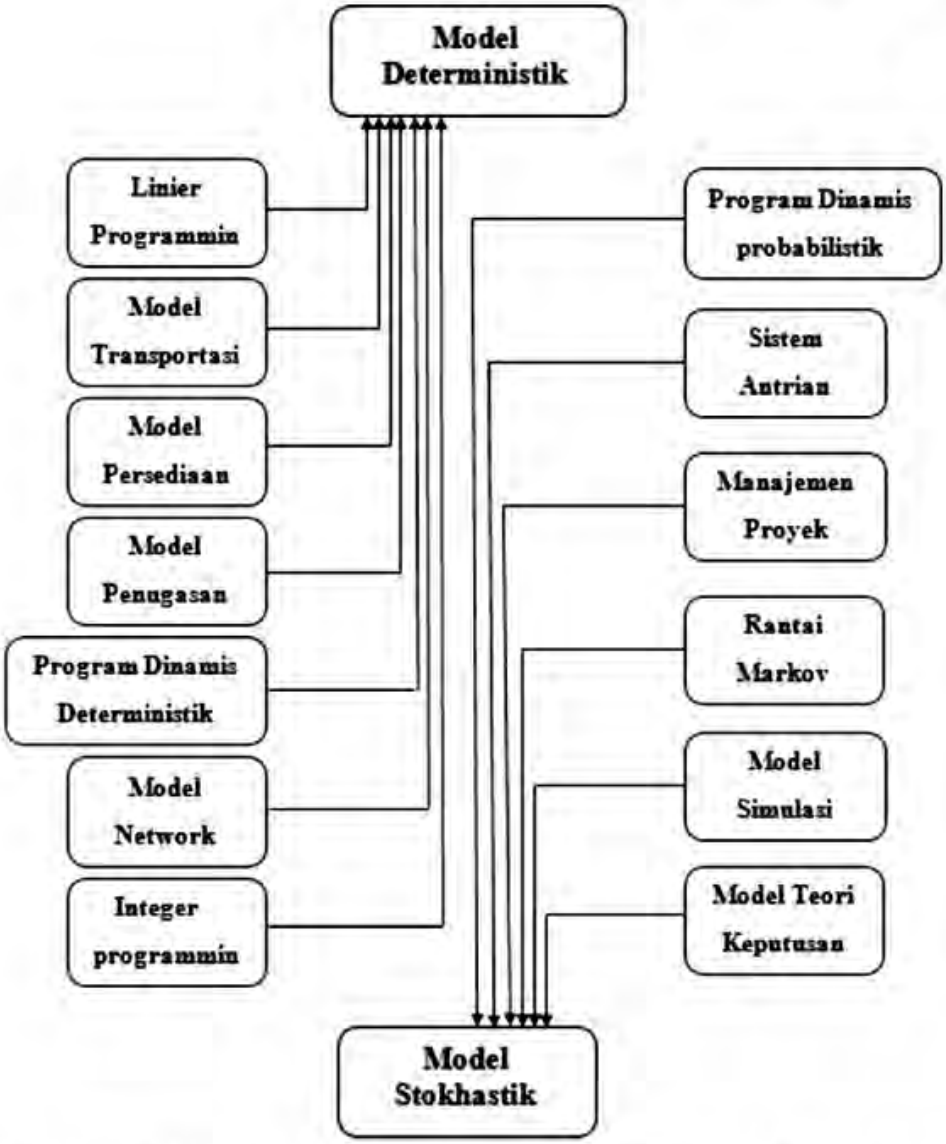
Tahap krusial berikutnya adalah dalam proses pemodelan adalah implementasi model. Model yang sudah divalidasi menunjukkan bahwa model tersebut telah siap untuk diaplikasikan dalam sistem sesungguhnya. Pada tahap ini juga bisa dilakukan bersamaan dengan analisa dampak implementasi model pada *current system*. Analisa dampak atau lebih dikenal sebagai analisa sensitivitas ini sangat penting untuk memberi informasi kepada pengambil keputusan dalam beberapa hal terkait pengaruh perubahan faktor atau variable yang terlibat kepada kondisi optimal implementasi model. Informasi perubahan dan dampak ini sangat penting karena seringkali terjadi faktor dan variable yang terlibat tersebut terkait dengan implikasi kebijakan di beberapa level yang berbeda, misalnya:

- Implikasi manajerial (*managerial implications*), pada pendekatan optimasi beberapa faktor atau variabel yang sering digunakan pengambil kebijakan untuk dilakukan analisa sensitivitas yang terkait dengan kebijakan manajemen level atas antara lain terkait dengan perubahan harga bahan baku, ketersediaan sumber daya dan harga jual. Perubahan-perubahan pada variabel atau faktor tersebut memerlukan keterlibatan pembuat kebijakan level atas.
- Implikasi operasional (*operational implications*). Pada model optimasi sistem industri, pada level manajemen lebih bawah atau level operasional, analisa sensitivitas ini penting dilakukan dengan melibatkan level manajemen operasional terkait dengan perubahan variabel atau faktor seperti batasan kapasitas atau jumlah dan kombinasi produksi yang biasanya perencanaannya dilakukan pada *short term horizon*.

Pada dasarnya pengambil keputusan akan melihat hasil optimal dari implementasi model yang dilakukan sebagai acuan namun mereka juga perlu diberi informasi yang selengkap mungkin dengan kemungkinan alternatif perubahan yang bisa dilakukan dan dampaknya terhadap kondisi optimal serta implikasinya terhadap kondisi operasional maupun manajerial.

Klasifikasi Model Matematis dalam Optimasi

Dalam model matematis yang menggunakan pendekatan optimasi dalam solusinya terdapat dua jenis model matematis, yakni model matematik deterministik dan model matematik stokhastik. Model optimasi yang berupa matematik determistik adalah jenis model matematis yang gejala-gejala dan parameternya dapat diukur dengan tingkat kepastian yang tinggi karena mempunyai peluang kejadian yang tetap dan tidak berubah. Beberapa model optimasi matematik deterministic antara lain problem transportasi, problem penugasan, model jaringan, model *transhipment problem* yang merupakan turunan dari problem utama linier programming. Bebarapa contoh problem model matematik deterministik serta klasifikasi dan perbedaannya dengan model matematis stokhastik bisa dilihat dalam tabel klasifikasi model matematis di bawah ini:



Gambar 2. 2 Klasifikasi Model Matematis Dalam Pendekatan Optimasi

Dalam Gambar 2.2 di atas, terlihat pengelompokkan beberapa jenis model matematis yang terkait dengan optimasi. Kata stochastik berasal dari kata Yunani stokhazesthai yang berarti membidik atau menebak, sehingga model stokhastik terkait dengan inputan atau parameter model yang hanya bisa dijelaskan dengan distribusi peluang dan perubahannya sangat tergantung dengan unsur waktu sehingga model stokhastik tidak bisa ditentukan fungsi atau formulasi matematis secara permanen atau pasti namun hanya bisa diprediksi berupa perkiraan yang nilainya belum pasti (*stochastic*). Menurut Stephanie (2018) dalam artikelnya

"*Statistics How To*" mendefinisikan stokhastik sebagai "*situasi dimana ketidakpastian hadir atau dengan kata lain, ini adalah model untuk proses yang memiliki semacam keacakan. Dalam kata sebenarnya, ketidakpastian adalah bagian dari kehidupan sehari-hari, jadi model stokastik secara harfiah bisa mewakili apapun, di mana hal tesebut berlawanan dengan model deterministik, yang memprediksi hasil dengan kepastian 100%. Sementara itu model stokastik kemungkinan akan menghasilkan hasil yang berbeda setiap kali model dijalankan*".

Model optimasi matematis stokhastik bisa dikelompokkan atas beberapa model matematis antara lain program dinamis probalilistik, model sistem antrian, model manajemen proyek, model rantai Markov, model simulasi dan model teori keputusan. Model-model stokhastik tersebut cenderung lebih sulit diselesaikan karena ada unsur ketidakpastian (*stochastic*) dibandingkan dengan penyelesaian model-model deterministik. Acapkali penyelesaian problem model stokhastik ini tidak berfungsi sebagaimana pendekatan optimasi yang memberikan solusi pasti namun hasil penyelesaian dengan menggunakan model stokhastik ini cenderung hanya memberikan informasi bagi para pengambil keputusan untuk memperbaiki keputusannya. Model rantai Markov dan model simulasi adalah dua contoh yang cukup jelas bahwa bukan solusi pasti yang diberikan informasi terkait perubahan kondisi sistem yang berubah seiring dengan perubahan dinamis waktu dan parameter di dalamnya yang terus berubah. Di dalam model *Markov chain* ditunjukkan bagaimana probabilitas sebuah event atau kejadian yang berubah terus dari waktu ke waktu sampai didapatkan sebuah kondisi *steady state* (kondisi tetap) pada satu titik waktu tertentu yang kemudian akan dipakai untuk memprediksi *state*/kondisi sesuai jumlah event di masa datang. Sehingga bisa dikatakan bahwa hasil prediksi tersebut sama sekali bukan sebuah solusi pasti, namun hanyalah sebuah perkiraan atau prediksi yang belum 100% pasti kebenarannya karena mengandung unsur probabilitas. Selanjutnya model simulasi juga memberikan cara kerja yang hampir sama.

Model simulasi merupakan model tiruan dari sistem yang sedang digambarkan lengkap dengan seluruh variabel, entitas dan parameter yang ada di *real system*. Alasan terbesar mengapa menggunakan model simulasi ini adalah sangat kompleksnya sistem yang akan dimodelkan termasuk perubahan continue atas parameter di dalamnya sehingga

akan sangat berbahaya dan mahal jika tidak dimodelkan dalam bentuk simulasi modelnya. Dalam perspektif optimasi, model simulasi ini bahkan tidak dimasukkan ke dalam model solutif yang memberikan soluasi pasti atas problem yang dihadapi, sehingga bukan solusi yang dihasilkan dari output model ini namun lebih pada memberikan alternatife/skenario atau eksperimen atas perubahan parameter di dalamnya dan memberikan informasi dampaknya pada keseluruhan sistem. Contoh paling jelas dari model simulasi ini adalah memodelkan sistem antrian kompleks di sebuah proses produksi manufaktur yang mempunyai sistem *multi channel* dan *multi-supply* dan *multi hierarcy flows* dimana solusi yang ditawarkan sistem sistem antrian sederhana tidak akan bisa menyelesaikan seluruh persoalan antrian. Proses looping dan *re-work* yang terjadi antar mesin dan divisi mempersulit solusi optimal dengan teori antrian sederhana yang biasanya memberikan soluasi berupa usulan penambahan server baru. Pada pemodelan simulasi ini juga ada tahap verifikasi dan validasi model sebelum diterapkan dalam kondisi sebenarnya. Pada proses verifikasi berfungsi untuk memeriksa dan memastikan penerjemahan model konseptual simulasi seperti algoritma, flowchart dan parameter-parameter di dalamnya telah berjalan dengan benar dan sesuai *current situation* sementara tahap validasi bertujuan untuk menjawab bahwa model yang didesign telah merepresentasikan kondisi nyata secara akurat yang biasanya dilakukan secara obyektif (statistik) atau subyektif (kepakaran) dengan membandingkan kondisi nyata dan model.

Model Optimasi Matematis Sederhana: Tanpa Kendala

Sebuah sistem persamaan yang dimodelkan secara matematis bisa digunakan sebagai salah satu cara untuk menyelesaikan sebuah permasalahan optimasi bisa dibedakan atas jumlah variabel yang terlibat di dalam persamaan tersebut serta bisa dibedakan atas ada tidaknya kendala (*constraint*). Untuk model matematis tanda kendala yang terdiri atas lebih dari satu variabel maka penyelesaian optimasinya akan lebih sulit dibandingkan dengan persamaan yang hanya terdiri atas satu variabel saja. Hal ini juga berlaku pada model sistem persamaan matematis yang terdapat persamaan tambahan sebagai syarat atau kendala yang jelas lebih sulit diselesaikan dibandingan dengan mode sistem persamaan matematis tanpa persamaan prasyarat atau kendala.

Jika suatu fungsi persamaan matematis lebih dari satu variabel tanpa menggunakan persamaan kendala untuk kasus optimasi kombinasi produksi diberikan sebagai fungsi $\pi = f(x;y)$ dengan fungsi matematis:

$$\pi = 12x+18y-2x^2-xy-2y^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots 1)$$

Dimana π = fungsi tujuan sebagai keuntungan produksi

x = output produksi produk 1

y = output produksi produk 2

Dalam sistem persamaan tersebut variabel x dan y disebut veraiable independen yang tidak terkait satu dengan yang lainnya serta tidak ada persamaan lain yang membatasi kedua variabel tersebut. Sebagai salah satu model optimasi, persamaan matematis dua variabel tanpa kendala di atas mengharapkan ada soluasi optimal nilai x dan y, dimana titik optimasi fungsi merupakan titik optimal bebas yang dicapai pada kondisi $\pi' = 0$ dimana π' adalah turunan pertama dari fungsi tersebut. Sehingga solusi optimal fungsi tersebut bisa didapatkan dengan menurunkan pada turunan pertama fungsi terhadap masing-masing variabel x dan y:

$$\delta\pi/\delta x=12-4x-y=0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 2)$$

$$\delta\pi/\delta y=18-x-4y=0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 3)$$

Dua sistem persamaan di atas bisa diselesaikan dengan menggunakan eliminasi dan substitusi didapatkan hasil sebagai berikut:

$$48 - 16x - 4y = 0$$

$$\underline{18 - x - 4y = 0} \; -$$

$$30-15x = 0$$

$$x = 2$$

Dengan mensubstitusikan nilai x ke dalam persamaan 2) atau 3) didapatkan nilai y = 4. Ini berarti bahwa produk 1 akan diproduksi sebesar 2 unit dan produk 2 akan dibuat sebanyak 4 unit. Nilai keuntungan produksi didapatkan dari memasukkan nilai x dan y tersebut ke dalam persamaan 1), sehingga keuntungannya akan menjadi 48.

Model sistem persamaan matematis diatas terlihat lebih mudah dicari nilai optimasinya karena tidak ada persamaan lain (kendala) yang

membatasi sistem persamaan tersebut. Apabila terdapat persamaan lain sebagai kendala maka diperlukan penyelesaian yang lebih panjang untuk mencari nilai optimalnya dengan beberapa pendekatan salah satunya adalah dengan linier programming yang akan dibahas di bab 3.

Optimasi dalam R

Software R punya package khusus yang menyediakan aplikasi sederahana untuk problem-problem model optimasi deterministik seperti linier programming, problem transportasi, problem penugasan, program bilangan integer dan program lain yang menjadi turunan dari linier programming. Obyek atau fungsi-fungsi optimasi di atas yang dikemas dalam bentuk package atau add-ins akan disimpan dalam memori program. Untuk mengetahui package apa saja yang ada di program R bisa dengan melihat dengan cara memasukkan perintah search() setelah prompt >.

```
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

Fungsi search merupakan fungsi tanpa argument yang hasil eksekusinya didapatkan pada setiap obyek yang diketikkan dengan mencari dan menelusuri ke "GlobalEnv"; "package:stats"; "package:graphics"; "package:grDevices"; "package:utils"; "package:datasets"; "package:methods"; "Autoloads"; " package:base". Untuk setiap input atau obyek yang dimasukkan ke dalam *console* namun tidak ditemukan diproses penelusuran akan diberi notifikasi *error*.

Fungsi-fungsi yang telah ada di dalam *console* bisa dilihat di dalam "package:'base'" dengan cara memasukkan perintah help setelah prompt >:

```
>help(package: 'base')
```

Hasilnya adalah R akan mencoba mengakses sebuah windows yang terkoneksi http yang berisi library atau dokumentasi semua fungsi atau perintah yang ada di package base R seperti tampilan gambar di bawah ini:

Gambar 2. 3 Library R

Untuk problem optimasi dari model matematis, misalnya menggunakan metode linier programming untuk persamaan tanpa kendala, R console memerlukan install package lpsove dan lpsolveAPI setiap kali akan menggunakannya. Perintah install ini diperlukan agar package yang belum *built-in* di R console bisa dimasukkan ke dalam library R untuk digunakan. Perintah atau command yang digunakan untuk menginstall adalah:

>install.package("lpSolve")

Sementara untuk mengintal package lpSolveAPI menggunakan perintah yang hampir sama:

>install.package("lpSolveAPI")

Perbedaaan antara kedua versi package linier programming di atas adalah tingkat kelengkapan problem yang bisa diselesaikan. lpSolve yang dirilis dalam versi terakhir 5.5.07 tahun 2007 menyediakan *high-level functions* untuk menyelesaikan problem linier atau integer program secara umum, *assignment problem* dan *transportation problems.* Seementara itu package lpSolveAPI versi 5.5.2.0 yang dirilis lebih baru yakni tahun 2010 memiliki jauh lebih banyak fungsi daripada lpSolve yang tingkat kesulitannya cukup tinggi.

Penyelesaian Model Optimasi Sederhana dengan R

Sebagai contoh, sekarang kita akan menentukan nilai x dan y dari model optimasi berikut : $f(x,y) = 20x + 18y - 2x^2 - xy - 2y^2$. Problem tersebut dapat diselesaikan dengan metode analisis numeric pada R. Berikut ini adalah syntax penyelesaian problem dengan R.

```
> f <- function(x){12*x[1]+18*x[2]-2*(x[1])^2-x[1]*x[2]-2*(x[2])^2}
> initial_x <- c(1, 5)
> x_optimal <- optim(initial_x, f, method="CG")
> x_max <- x_optimal$par
> x_max
```

Dengan analisis numeric tersebut diperoleh hasil nilai x = -2.184494e+15 dan y = 1.503576e+15 dengan output :

```
[1] -2.184494e+15  1.503576e+15
```

Sehingga dapat disimpulkan bahwa kondisi optimal maksimum fungsi f diperoleh saat nilai x sama dengan -2.184494e+15 dan nilai y sama dengan 1.503576e+15.

Syntax yang digunakan dalam menyelesaikan permasalahan modeloptimasi sederhana tersebut terdiri dari beberapa argumen. Secara detail masing masing baris argumen dapat diartikan sebagai berikut :

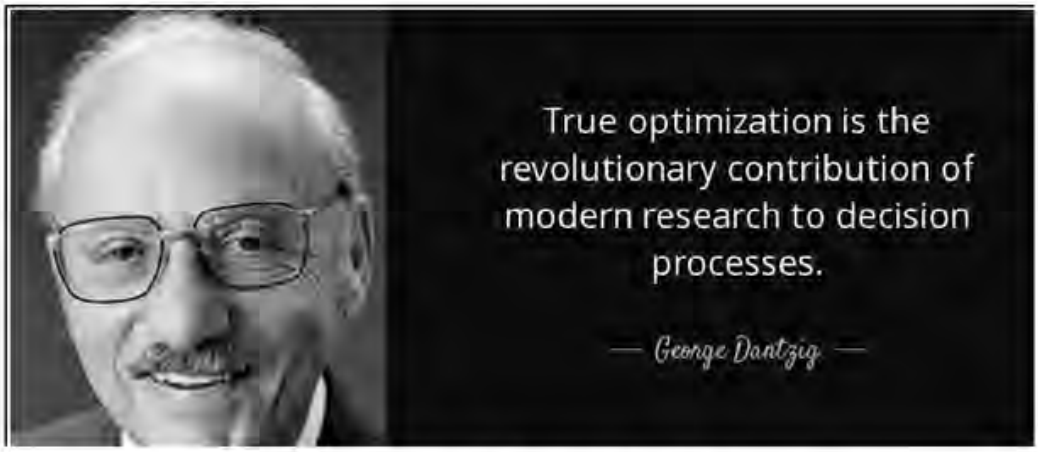
a. f : adalah fungsi yang akan dioptimalkan (minimum atau maksimum). Pada contoh diatas dituliskan function(x)..., untuk membedakan x dan y atau x1 dan x2 maka digunakan [1] dan [2] dibelakang variabel x.
b. initial_x : adalah nilai awal untuk variabel yang akan dioptimalkan.
c. method : adalah metode penyelesaian yang digunakan, pada contoh ini digunakan metode default yaitu CG. Metode lainnya dapat dilihat dengan lebih detail pada manual R atau library R.

BAB 3
PEMROGRAMAN LINIER

Karakteristik Pemrograman Linier
Formulasi Pemrograman Linier
Perubahan Dari Elemen-Elemen Pemrograman Linier
Mengubah Pemrograman Linier Kedalam Bentuk Standar
Penyelesaian Model Pemrograman Linier
Penyelesaian Model Pemrograman Linier: Metode Grafik
Masalah-Masalah Khusus dalam Pemrograman Linier
Langkah-Langkah Penyelesaian Metode Grafik
Penyelesaian Pemrograman Linier dengan R
Analisis Sensitifitas Secara Grafis
Pemrograman Linier Dengan Metode Simpleks

PEMROGRAMAN LINIER

Pemrogaman linier (*Linear Programming*) adalah salah satu teknik yang sangat sering digunakan dalam metode optimisasi. Metode ini dibuat pada tahun 1937 ketika Leonid Kantorovich mempublikasikan penelitiannya yang berjudul "*A new method of solving some classes of extremal problems*". Kantorovich mengembangkan pemrograman linier sebagai metode untuk merencanakan pengiriman dan pengembalian dalam pemesanan untuk mengoptimalkan biaya yang dikeluarkan tentara ketika itu dan memberikan kerugian bagi musuh. Metode ini tetap dirahasiakan hingga tahun 1947, ketika George B. Dantzig yang sekarang disebut Bapak Linear Programming mempublikasikan metode simpleks untuk menyelesaikan teori dualitas dalam konteks analisis matematis dari teori permainan (*Game Theory*) di dalam sebuah literatur yang berjudul "*Linear Programming and Extension*".



Gambar 3. 1 George B. Dantzig dan Kutipannya

Setelah itu, pemrograman linier berkembang pesat sekali, semula ditujukan untuk bidang militer dengan mengaplikasikannya dalam merencanakan startegi petempuran di medan perang) maupun dalam aspek bisnis dengan fokus pada proses maksimasi keuntungan, mengurangi kerugian dan lain sebagainya). Saat ini penggunaanya sudah sedemikian cepat dan luas mencakup problem perancanan dan pengendalian pembangunan di bidang finansial atau keuangan "*allocation of investment*" dan sektor-sektor perekonomian terkait kebijakan dalam peningkatan devisa, ekspor dan kebijakan bisnis nasional.

Salah satu alasan terkenalnya pemrograman linier adalah memungkinkan pemodelan berbagai macam situasi dengan kerangka sederhana. Selanjutnya, pemrograman linier relatif mudah untuk diselesaikan. Metode simpleks memungkinkan untuk menyelesaikan pemrograman linier paling efisien, dan metode *inner-point* dari Karmarkar memungkinkan penyelesaian paling efisien dari beberapa jenis pemrograman linier.

Pemrograman linier ini akan sangat bagus jika dalam kasus advanced, seperti penyelesaian *integer* dan *mixed integer linear programming*. Pada model ini, variabel-variabel keputusan adalah berupa bilangan bulat. Untuk model ini, metode yang terkenal adalah *branch and bound* yang diperkenalkan oleh Land dan Doig. Beberapa algoritma muncul kemudian, seperti misalnya metode *cutting plane*. Metode ini dan pengembangan dari perhitungannya, telah berkembang secara pesat untuk pemrograman linier.

Beberapa contoh penerapan pemrograman linier adalah sebagai berikut.

1. Rencana Produksi dan Persediaan

   Pada kasus seperti ini, kuantitas produk yang akan dibuat bisa ditentukan pada setiap periodenya untuk digunakan untuk memenuhi permintaan dengan tujuan meminimumkan biaya persediaan, penyewaan fasilitas, lembur dan subkontrak.

2. Rencana Produksi dengan beberapa jenis produk

   Dalam hal ini kita 98an menentukan berapa unit setiap produk yang akan dibuat agar memaksimumkan keuntungan dengan memperthatikan kemampuan permintaan dan kendala produksi.

3. Perencanaan Investasi

   Pada kasus perencanaan investasi, kita akan menentukan berapa jumlah funding yang akan diinvestasikan untuk bisa memaksimalkan nilai pengembalian atau "*return of investment*" (ROI) atau nilai sekarang "*net present value*" (NPV) dengan mempertimbangkan ketersediaan anggaran yang tersedia untuk diinvestasikan dan ketentuan dari tiap beberapa keputusan inventasi.

Penerapan dari pemrograman linier tidak hanya terbatas pada contoh di atas. Pemrograman linier juga dapat diterapkan pada masalah perencanaan lokasi, *break event point* dari sebuah investasi, penjadwalan tenaga kerja dan mesin, penugasan, pemanfaatan lahan pertanian, muatan truk dan sebagainya.

Pada bagian ini pengenalan akan program linier akan dibahas secara mendalam, beserta beberapa formulasi sederhana. Kita juga akan membahas mengenai pemanfaatan software R untuk menyelesaikan pemrograman linier, menggunakan penerapan dari libraries bernama *lpSolve*.

Karakteristik Pemrograman Linier

Karakteristik yang dimiliki LP adalah berusaha mendapatkan maksimasi atau minimasi, baik berupa maksimasi keuntungan (profit) atau maksimasi *Rate of investment* dan sebagainya maupun berupa minimasi biaya (*costs*) melalui langkah-langkah penyelesaian yang sangat khas dan sistematis matematis . Langkah-langkah penyelesaian problem LP sebagai berikut:

Langkah 1 : identifikasikan tujuan utama masalah yang dihadapi, kemudian tentukan apa keputusan yang akan diambil dan faktor apa yang menjadi kendalanya.

Langkah pertama dalam LP ini sangat krusial untuk dipahami oleh problem solver mennggunakan LP, karena memahami permasalahn yang sedang dihadapi sangat menentukan keputusan yang akan dibuat. Misalnya saat berhadapan dengan permasalahan kombinasi produksi maka kita sebenarnya sedang berhadapan dengan pengambilan keputusan mix combination produksi berbasis keuntungan produksi yang maksimal. Begitu juga sebaliknya saat kita sedang menyelesaikan problem distribusi dan transportasi maka kita harus paham bahwa tujuan akhir dari penyelesain problem ini dengan menggunakan LP adalah menentukan alokasi distribusi dari sumber ke tujuan dengan berbasis pada total biaya distribusi minimal.

Langkah 2 : dari analisa langkah satu, definisikanlah variabel keputusan.

Setelah memahami problem yang sedang dihadapi seperti pada langkah pertama, langkah berikutnya dalam penyelesaian problem dengan menggunakan LP adalah mendefinisikan variabel keputusan. Langkah pendefinisan variabel keputusan diperlukan untuk mensinkronkan variabel keputusan dan jawaban akhir atau keputusan yang diambil. Definisi variabel keputusan ini dilakukan dengan membuat formulasi matematis dari variabel keputusan yang sersurat di problem yang sedang dihadapi yang terkadang sangat sulit diidentifikasi dan dibuat formulasi matematisnya. Definisi variabel keputusan secara matematis ini sangat penting karena definisi matematis ini akan dipakai sebagai variabel di formulasi matematis langkah berikutnya misalnya di formulasi matematis fungsi tujuan dan formulasi matematis fungsi kendala atau batasan.

Langkah 3 : Dari analisa langkah 2, tentukan tujuan yang bisa berbentuk "maksimum" atau "minimum".

Langkah berikutnya dalam penyelesaian problem dengan LP adalah membentuk formulasi fungsi tujuan. Formulasi fungsi tujuan menunjukkan tujuan utama yang jadi pertimbangan utama dalam menyelesaikan sistem persamaan dalam linier programming. Pada dasarnya ahnya ada dua jenis fungsi tujuan kalau dilihat dari orientasi keputusan yang diambil, yakni fungsi tujuan maksimasi dan fungsi tujuan minimasi.

Langkah 4 : Tentukan faktor kendala yang bisa berbentuk [≤] (biasanya diinformasikan sebagai "paling banyak" atau maksimum atau [≥] yang biasanya diinformasikan dengan "paling sedikit" atau "minimum" atau non negativity, ataupun bisa berbentuk [=].

Langkah ini sangat menentukan karena akan terkait dengan proses penentuan daerah kelayakan (*feasible area*) yang akan mempengaruhi penentuan titik optimal. Untuk kasus-kasus linier programming tertentu seperti problem transportasi, beberapa kendala (*contraint*) biasanya adalah kapasitas gudang pengiriman yang menggunakan tanda (≤) dan kendala permintaan (*demand*) yang kendalanya [=]. Sedangkan problem Lp lain seperti kombinasi prouksi (*production mix*) beberapa kendala biasanya adalah kapasitas jam tenaga kerja dan kapasitas bahan baku yang menggunakan tanda [≤] serta kadang ada kendala minimal unit produksi yang diinginkan dengan tanda [≥].

Secara teoritis, dalam proses pemrograman linier mempunyai beberapa ciri khas, sifat maupun karakteristik dalam memodelkan maupun dalam proses pencarian solusi yang perlu diperhatikan. Ciri khas dan karakteristik ini juga disebut sebagai asumsi dasar linier program seperti sifat liniearity, proporsionality, additivitas, divisbility, kepastian, serta sifat tidak negativ atau non negativity. Penjelasan lebih lanjut dan lengkap terkait ciri khas, sifat dan karakteristik dasar dari linier programming diterangkan dalam penjelasan di bawah ini.

Linearity dalam programa linier bisa dilihat dan diidentifikasi dengan memakai beberapa teknik. Misalnya dengan pendekatan statistik kita bisa melihat kelinieritasan sebuah fungsi dengan memakai grafik atau diagram pencar. Bahkan bisa juga dilakukan sebuah uji hipotesis untuk mengetahui apakah sebuah fungsi tersebut linier atau bukan. Selain itu, linearity sebuah fungsi matematis secara teknis bisa diukur oleh sifat atau ciri khas lain dari programa linier dari sifat proporsionalitas, additivitas, divisibilitas dan seberapa pasti/*robust* dari sebuah fungsi tujuan atau *objective function* dan kendala atau konstraint.

Karakter proporsionalitas dari sebuah fungsi programa linier dapat ditentukan, jika ditemukan kontribusi dari setiap variabel yang terlibat dalam sistem programa linier terutama di objective function atau fungsi tujuan dan pemakaian kendala atau sumber daya yang membatasi secara proporsional terhadap nilai variabel. Misalnya, apabila terjadi per unit harga dari sebuah produk (variabel keputusan) selalu sama berapapun

jumlah material, bahan baku atau komponen produk yang dibeli, maka sifat proporsionalitas tersebut akan berlaku dan terpenuhi. Dengan kata lain, sifat proporsionalitas akan tidak berlaku jika terjadi sebaliknya misalnya pembelian material, bahan baku atau komponen produk yang meningkat atau lebih banyak kuantitasnya akan mendapatkan harga per unit yang lebih murah atau mendapatkan diskon. Kesimpulannya, jika penggunaan sumber daya seperti material, bahan baku dan komponen produk yang diperlukan per unitnya sangat tergantung perubahannya tergantung dan dipengaruhi dari jumlah yang diproduksi, maka sifat proporsionalitas dari sebuah fungsi programma linier tidak dipenuhi.

Karakteristik additivitas dalam program linier berasumsi bahwa berbagai aktivitas dalam sistem program linier tidak saling membentuk perkalian silang di dalam model. Karakteristik ini berlaku dalam programma linier pada fungsi tujuan dan fungsi sumber daya di mana fungsi tujuan dalam sebuah sistem prigram linier merupakan penambahan secara langsung dari setiap variabel-variabel keputusan yang terlibat. Khusus untuk fungsi pembatas atau kendala dalam programa linier, karakteristik additivitas bisa dipenuhi apabila nilai ruas kanan dari persamaan fungsi kendala merupakan jumlah atau total pemakaian tiap-tiap variabel keputusan yang terlibat. Dengan kata lain, apabila dalam sebuah sistem programma linier terdapat jumlah tertentu variabel keputusan yang mewakili jumlah produk tertentu dimana saat terjadi peningkatan jumlah atau kuantitas produksi dari salah satu atau beberapa variabel keputusan (produk) akan mempengaruhi (menaikkan atau menurunkan) jumlah atau kuantitas produksi beberapa variabel keputusan atau produk lain dalam sebuah proses atau periode yang sama maka karakteristik additive tersebut berarti tidak terpenuhi.

Karakteristik divisibilitas dalam programma linier punya arti bahwa unit variabel keputusan yang menjadi jawaban dalam sebuah problem linier programming bisa dibagi ke bentuk level fraksional jadi akan muncul nilai hasil atau variabel keputusan yang tidak bulat atau tidak integer.

Karekteristik certainty atau tingkat kepastian dalam programa linier mengisyaratkan bahwa semua parameter di dalam model persamaan linier merupakan konstana yang tetap besarnya yang tidak berubah atas waktu. Sehingga dalam sebuah sistem program linier nilai koefisien yang terdapat di model fungsi tujuan serta koefisien yang terdapat di dalam

model fungsi pembatas merupakan niali yang selalu pasti dan bukan nilai yang tidak pasti dengan peluang tertentu.

Karakteristik non-negativity atau lebih dikenal sebagai sifat tidak negatif di dalam programa linier yang menunjukkan bahwa variabel keputusan atau semua jawaban yang akan dicari nanti merupakan nilai jawaban yang harus positif atau nol (0) atau tidak negatif. Sifat ini merupakan sifat yang paling dasar dan utama dari model programma linier sehingga seringkali fungsi non-negativity ini tidak ditulis oleh para pemodel karena sudah dianggap dipahami oleh semua orang.

Dalam kenyataan riil-nya dalam menyelesaikan problem linier programming keenam karakteristik dasar tersebut di atas tidak selalu dapat dipenuhi, sehingga sebuah analisa yang dikenal dengan analisa sensitivitas (terhadap solusi optimal yang telah didapatkan) kadang diperlukan untuk memastikan terpenuhinya keenam karakteristik tersebut dalam model linier. Keenam asumsi (sifat) ini dalam dunia nyata tidak selalu dapat dipenuhi. Pembahasan teori analisa sensitivitas akan dibahas secara khusus di sub bab berikutnya.

Formulasi Pemrograman Linier

Struktur dari Model Pemrograman Linier

Dalam sub bab sebelumnya telah dijelaskan bahwa programa linier adaah sebuah metode yang bertujuan untuk mencari nilai optimal (baik itu meminimalkan ataupun memaksimalkan solusi) dari sebuah fungsi tujuan atau *objective function* yang merupakan sebuah persamaan linier. Selain itu juga diketahui bahwa variabel-variabel yang terlibat di dalam model juga berupa nilai linier dan hanya dapat berupa didefiniskan dengan batasan atau kendala yang juga berbentuk linier. Dengan kata lain, semua fungsi persamaan matematis di dalam programa linier baik fungsi tujuan serta batasan-batasan yang terlibat di dalam programa linier semuanya berbentuk persamaan linier.

Pada dasarnya, problem programa linier merupakan problem yang dipakai untuk mencari besarnya nilai dari setiap variable keputusan dengan harapan nilai fungsi tujuan atau *objective function* menghasilkan nilai yang optimal. Nilai optimal ini sangat tergantung dari jenis problem yang sedang dihadapi. Jika problemnya adalah meminimalkan biaya, maka nilai minimal fungsi tujuan yang dicari atau saat problemnya adalah memaksimalkan keuntungan maka fungsi tujuan yang diharapkan

adalah maksimal. Kondisi optimal (bisa berupa maksimal atau minimal) dari fungsi tujuan yang dicari tentunya dengan mempertimbangkan kendala atau batasan yang berlaku yang akan mencadi inputan dalam proses pencarian solusi. Kendala atau batasan yang terlibat dalam programa linier harus dinayatakan dalam bentuk persamaan linier (*linear*) meskipun dalam proses awalnya bentuk persamaannya berupa ketidaksamaan (*inequality*). Hal ini biasa disebut sebagai *linear inequality* dalam programa linier.

Dalam penyelesaian problem programa linier, yang pertama kali perlu dilakukan yaitu dengan mempelajari dan memahami sistem permasalahan yang sedang dihadapi dengan baik yang dilanjutkan dengan melakukan pengembangan permasalahan dengan jelas dan detail. Pemahaman permasalahan yang sedang dihadapi ini termasuk di dalamnya adalah memahami pernyataan terkait fungsi tujuan yang dicari, batasan sumber daya yang dimiliki, hubungan antara masing-masing batasan, periode yang terlibat dalam permasalahn yang dihadapai serta ada tidaknya alternatif yang mungkin bisa dilakukan di dalam permasalahan tersebut dan aspek-aspek lain yang mungkin ada. Pada langkah awal ini, pemahaman akan fungsi tujuan dan pemahaman akan batasan atau kendala yang terlibat akan menjadi aspek yang paling krusial sebelum masuk ke tahap formulasi matematis permasalahan yang sedang dihadapi yang akan dilakukan dalam tahap berikutnya.

Pada tahap selanjutnya yang perlu dilakukan setelah proses pemahaman permasalahan optimasi yang sedang dihadapi adalah memodelkan secara matematis permasalahan menjadi sebuah bentuk formula matematis. Namun sebelumnya perlu dipikirkan untuk membuat bentuk permasalahan kompleks yang sedang dihadapi menjadi bentuk yang mudah dipahami dan simpel. Bentuk atau model disini bisa diberupa model berbentuk tabel atau bentuk lain yang bisa merepresentasikan permasalahan yang dihadapi agar mudah dipahami. Langkah ini akan memudahkan pemodel membentuk model matematis. Ketepatan penyederhanaan ini akan mempengaruhi ketepatan membuat model matematis dari permasalahan yang sedang dihadapi. Selanjutkanya bisa dilakukan dengan memformulasikan permasalahn ke dalam bentuk matematis optimasi. Pada intinya, membangun dan membentuk model optimasi matematis yang menggambarkan kondisi permasalahan yang sebenarnya merupakan sebuah pendekatan operation

research. Permasalahan yang seringkali berupa *case study* berbentuk soal cerita dengan pendekatan *operation research* akan coba diterjemahkan kedalam bentuk model matematis. Model matematis yang dibuat tersebut seharusnya merupakan bentuk lain atau representasi secara kuantitatif (numeris) dari fungsi tujuan dan batasan sumber daya yang dimiliki yang membatasi proses pencarian nilai optimal (solusi akhir) dari variabel keputusan. Secara detail, memodelkan matematis dari permasalahan yang sedang dihadapi secara optimal bisa dikategorikan ke dalam dua tahap.

Pada tahap pertama yaitu memodelkan secara matematis fungsi tujuan optimasi. Pada proses ini, model matematis yang dibentuk pada fungsi tujuan selalu berbentuk persamaan (=). Bantuk persamaan dari model matematis fungsi tujuan selalu digunakan di syaratkan karena adanya solusi optimal yang diharapkan berada pada satu titik saja. Dengan kata lain satu titik optimal yang dicari adalah nilai tertinggi (maksimal) atau nilai tersendah (minimal). Meskipun diperkembangan programa linier memungkinkan terjadi terdapat lebih dari satu tujuan (*multi goal programming*) akan tetapi dalam kesempatan ini kita hanya membahas permasalahan optimasi dengan *single goal* (satu tujuan) saja.

Pada tahap kedua yaitu merupakan proses memodelkan secara matematis dari representasi batasan sumber daya. Fungi matematis dari batasan sumber daya ini tidak selalu berbentuk persamaan (=) namun bisa juga berbentuk pertidaksamaan (≤ atau ≥). Pada beberapa kesempatan, fungsi matematis dari batasan sumber daya ini sering disebut sebagai constraint atau kendala. Di dalam model matematis dari fungsi kendala ada dua jenis konstanta atau koefisien yaitu konstanta ruas kiri dan konstanta ruas kanan. Konstanta atau koefisien yang berada di fungsi kendala in juga terdapat di fungsi tujuan meruapkan parameter dari model matematis kedua fungsi. Inilah yang membedakan antara model matematis dengan model verbal (soal cerita) dimana pendeskripsian nilai dari masing-masing variabel di setiap model matematis bisa dengan jelas direpresentasikan. Keuntungan lainnya saat memodelkan permasalahan ke dalam model matematis adalah penggambaran masalah yang kadang sangat kompleks menjadi lebih sederhana dan simple. Sebagai dampaknya, proses pemahaman dari struktur permasalahan yang rumit akan lebih mudah dan lebih bisa menggambarkan hubungan antara beberapa variabel dengan variabel lain atau hubungan fungsi satu dengan fungsi yang lainnya. Selain itu,

dengan memodelkan permasalahan menjadi model matematis terutama permasalahan programa linier, hubungan simultan atau wholistic atau hubungan keseluruhan secara bersama-sama menajdi lebih terjamin. Satu hal yang lebih penting dari memodelkan matematis dari permasalahan yang sedang dihadapi adalah menyangkut penyelesaian optimasi tingkat lanjut dengan memakai teknik komputasi, bahasa pemrograman atau coding yang jelas akan lebih mudah dikerjakan. Misalnya saat menggunakan teknik kompoter seperti LINGO, MATLAB, QS WIN, TORA, R atau bahasa pemrograman yang lainnya maka model matematis yang telah dibuat dengan prinsip dasar linier programming akan menjadi jembatan atau penghubung yang sangat penting dalam proses *solution searching*.

Meskipun model matematis dari linier programming telah dijelaskan panjang lebar memberi kemudahan dalam proses pencarian solusi, namun tetap saja masih ada titik kelemahan yang bisa dijumpai saat memodelkan secara matematis sebuah permasalahan verbal. Misalnya saja, tidak semua karakter dari sebuah sistem rill yang kompleks bisa dimodel matematiskan sehingga seringkali terjadi model matematis yang dibuat tidak merepresentasikan *real system* yang benar. Sehingga sebagai akibatnya, seringkkali sulit ditemukan soluasi optimalnya atau jika ditemukan hasil akhirnya bukan merupakan solusi yang tepat. Selain itu, untuk tingkat kompleksitas permasalahan yang tinggi, model matematis yang tidak akuran akan menyebabkan proses *bridging* pada teknik komputasi tidak bisa diproses.

Berikut adalah bentuk umum formulasi dari pemrograman linier dengan fungsi tujuan berbentuk maxsimize (max Z).

Objective function (fungsi tujuan):

$$MAX\ Z = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$$

Constraint atau Fungsi kendala (subject to (s.t.) atau batasan sumber daya)

$$a11x1 + a12x2 + \ldots + a1nxn \le b1$$
$$a21x1 + a22x2 + \ldots + a2nxn \le b2$$
$$\ldots$$
$$am1x1 + am2x2 + \ldots + amnxn \le bm$$
$$x1, x2, \ldots, xn \ge 0$$

Model di atas mempunyai elemen-element sebagai berikut:

1. Sebuah fungsi tujuan dari sebanyak *n* variabel keputusan *xj*. Variabel keputusan dipengaruhi oleh koefisien biaya *cj*.
2. Sebuah himpunan dari batasan m, yang mana terdiri dari kombinasi linier dari variabel yang dipengaruhi oleh koefisien *aij* yang mempunyai nilai lebih kecil sama dengan sisi kanan (*right hand side*) dari nilai *bi* (batasan dengan tanda "lebih besar atau sama dengan" atau "sama dengan" juga ada).
3. Batasan dari variabel keputusan. Dalam kasus ini, seluruh variabel keputusan adalah bilangan non-negatif (variabel keputusan >=0).

Batasan pada pemrograman linier didefinisikan oleh *feasible region*, yang mana adalah nilai-nilai yang memenuhi seluruh konstanta yang ada. Untuk pemrograman linier dari sejumlah n variabel, *feasible region* adalah sebuah n-dimensi dari polytope cembung. Misalnya, untuk n = 2, *feasible region*-nya adalah poligon cembung.

Formulasi LP yang ditampilkan di atas bisa dirumuskan menjadi sebuah matriks seperti berikut ini (huruf besar tebal adalah matriks dan huruf kecil tebal adalah kolom vektor)

$$MAX\ z = \boldsymbol{c}'\boldsymbol{x}$$
$$s.t. \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$
$$\boldsymbol{x} \geq 0$$

Menggunakan penulisan matriks yang sama, kita dapat menulis bentuk pemrograman linier dengan fungsi tujuan minimum.

$$MIN\ z = \boldsymbol{c}'\boldsymbol{x}$$
$$s.t. \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$$
$$\boldsymbol{x} \geq 0$$

Cara lain untuk menyatakan sebuah model pemrograman linier adalah dalam bentuk standar. Bentuk ini dibutuhkan untuk menerapkan metode simpleks untuk menyelesaikan pemrograman linier. Kita dapat menggunakan OPT untuk menyatakan bahwa bentuk ini dapat didefinisikan kedalam model maksimum atau minimum.

$$OPT\ z = \boldsymbol{c}'\boldsymbol{x}$$
$$s.t. \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$
$$\boldsymbol{x} \geq 0$$

Contoh Sederhana dari Model Pemrograman Linier (contoh 1)

PT. OPTIMUMM menjual dua jenis produk, yaitu produk 1 dan produk 2. Satu ton dari produk 1 menghabiskan 30 jam kerja, dan satu ton dari produk 2 menghabiskan 20 jam kerja. Perusahaan ini hanya mempunyai maksimum 2700 jam kerja. Begitu juga dengan jam kerja mesin, satu ton produk 1 dan 2 menghabiskan masing-masing 5 dan 10 jam kerja mesin. Dan hanya ada 850 jam kerja mesin yang tersedia.

Satu ton dari produk 1 menghasilkan keuntungan Rp 20 juta, dan produk 2 menghasilkan keuntungan 60 juta untuk setiap ton yang terjual. Untuk alasan teknis, perusahaan minimal harus memproduksi minimal 95 ton untuk total kedua produk.

Untuk kasus ini, kita perlu mengetahui berapa ton produk 1 dan 2 yang harus diproduksi untuk memaksimalkan keuntungan. Untuk menyelesaikannya, dapat diterapkan model pemrograman linier. Pertama, kita perlu untuk mendefinisikan variabel keputusan. Pada kasus ini, kita mempunyai:

- P1 adalah jumlah produksi (dalam ton) dan jumlah yang terjual dari produk 1
- P2 adalah jumlah produksi (dalam ton) dan jumlah yang terjual dari produk 2

Koefisien biaya dari variabel ini adalah 20 dan 60. Kemudian, fungsi tujuan didefinisikan dengan mengalikan antara variabel keputusan dengan koefisien biaya masing-masing.

Kendala pada model pemrograman linier ini adalah:

1. Batasan dari jam tenaga kerja yang membuat jumlah jam kerja yang digunakan untuk produk 1 dan 2, yang sama dengan 30P1 + 20P2 adalah kurang dari atau sama dengan 2700 jam.
2. Sama dengan batasan jam tenaga kerja, batasan dari total jam kerja mesin 5P1 + 10P2 adalah kurang dari atau sama dengan 850 jam.
3. Batasan dari jumlah produksi minimal, yang membuat total ton produksi yang dijual untuk P1 + P2 akan bernilai lebih dari atau sama dengan 95.

Dengan penjabaran itu bisa kita susun formula sederhana dari kasus ini, berikut variabel keputusan adalah non-negatif, maka model pemrograman linier untuk memaksimalkan keuntungan adalah

Fungsi tujuan (memaksimalkan keuntungan)

$$MAX\ z = 20P1 + 60P2$$

Fungsi kendala

Kendala jam tenaga kerja

$$30P1 + 20P2 \leq 2700$$

Kendala jam kerja mesin

$$5P1 + 10P2 \leq 850$$

Kendala jumlah produksi minimal

$$P1 + P2 \geq 95$$

Kendala variabel non negatif

$$P1 \geq 0, P2 \geq 0$$

Masalah Transportasi (contoh 2)

Misalnya sebuah masalah transportasi yang terdiri dari 2 daerah asal pengiriman (misalnya pabrik), kita sebut a dan b, serta 3 tujuan (misalnya gudang), kita sebut dengan 1, 2 dan 3. Pada tabel 1 ditampilkan biaya transportasi *cij* untuk satu unit dari asal i ke tujuan j, dan kapasitas maksimal dari asal dan kebutuhan permintaan untuk tujuan. Kita perlu mengetahui bagaimana kita bisa memenuhi permintaan yang dibutuhkan tujuan dengan biaya yang minimal.

Tabel 3. 1 Contoh Masalah Transportasi

Pabrik/Gudang	1	2	3	Kapasitas
A	8	6	3	70
B	2	4	9	40
Permintaan	40	35	25	

Tabel ini dapat dimodelkan menjadi sebuah pemrograman linier sebagai berikut:

1. Variabel keputusan dalam bentuk *xij* menggambarkan berapa unit yang dikirim dari pabrik i ke gudang j.

2. Sebuah fungsi tujuan dengan koefisien biaya sama dengan *cij*
3. Dua jenis kendala: sebuah kendala dengan tanda "kurang dari atau sama dengan" untuk setiap pabrik, dibatasi oleh unit yang dikirim, dan sebuah kendala dengan tanda "lebih besar atau sama dengan" yang menggambarkan permintaan untuk setiap tujuan yang harus dipenuhi.

Pemodelan dari model tersebut adalah sebagai berikut:

Fungsi tujuan (meminimalkan biaya pengiriman)

$$MIN\ z = 8xa1 + 6xa2 + 3xa3 + 2xb1 + 4xb2 + 9xb3$$

Fungsi kendala

Kendala kapasitas pabrik a

$$xa1 + xa2 + xa3 \leq 70$$

Kendala kapasitas pabrik b

$$xb1 + xb2 + xb3 \leq 40$$

Kendala permintaan gudang 1

$$xa1 + xb1 \geq 40$$

Kendala permintaan gudang 2

$$xa2 + xb2 \geq 35$$

Kendala permintaan gudang 3

$$xa3 + xb3 \geq 25$$

Kendala variabel non negatif

$$xij \geq 0$$

Perubahan Dari Elemen-Elemen Pemrograman Linier

Sangat mudah untuk mengubah fungsi tujuan dari pemrograman linier. Sebuah masalah MAX dapat diubah menjadi MIN (dan sebaliknya), dengan mengganti tanda dari koefisien biaya:

$$MIN\ z = c'x \Leftrightarrow MAX\ z' = -c'x$$

Ketidaksamaan batasan dapat diubah dengan mengganti tanda dari seluruh batasan:

$$ai1x1 + \ldots + ainxn \leq bi \Leftrightarrow -ai1x1 - \ldots -ainxn \geq -bi$$

Ketidaksamaan batasan dapat juga diubah menjadi persamaan dengan menambahkan variabel non-negatif:

$$ai1x1 + ... + ainxn \leq bi \Rightarrow ai1x1 + ... + ainxn + si = bi$$
$$ak1x1 + ... + aknxn \geq bk \Rightarrow ak1x1 + ... + aknxn - ek = bk$$
$$si \geq 0, ek \geq 0$$

Batasan pertama di atas yang berbentuk pertidaksamaan (<=) diubah menjadi sama dengan dengan cara menambahkan sebuah variable semu yang bernama variabel *slack* si, dan batasan lebih besar sama dengan variabel *surplus* ek. Jika batasan asli harus dipertahankan, maka kedua tipe variabel tersebut haruslah berupa variabel non-negatif.

Terakhir, variabel keputusan dapat diubah juga. Sebuah variabel non positif x, dapat diganti dengan variabel non negatif x', menjadi x'I = -xi. Sebuah variabel yang tidak dibatasi pada tanda xk dapat diganti dengan 2 variabel non negatif x'k, x''k dengan xk= x'k – x''k.

Mengubah Pemrograman Linier Ke dalam Bentuk Standar

Biasanya, untuk mengubah model pemrograman linier adalah dengan cara mengubah seluruh batasan ke dalam persamaan dengan menambahkan variabel *slack* dan *surplus*. Untuk mudahnya, sebuah model didefinisikan sebelumnya dapat diubah menjadi bentuk standar seperti berikut:

$$MAX\ z = 20P1 + 60P2$$
$$s.t.WH)\ 30P1 + 20P2 + hW = 2700$$
$$MH\ 5P1 + 10P2 + hM = 850$$
$$PM)\ P1 + P2 - eP = 95$$
$$P1,\ P2,\ hW,\ hM,\ eP \geq 0$$

Dimana hw dan hm sama dengan jam tenaga kerja dan mesin. Dengan catatan variabel *slack* dan *surplus* harus berupa variabel non negatif.

Penyelesaian Model Pemrograman Linier

Prosedur yang sering digunakan untuk menyelesaikan masalah pemrograman linier adalah algoritma simpleks, yang dikembangkan George Bernard Dantzig pada tahun 1947. Pendekatan ini mengambil keuntungan dari fakta bahwa nilai optimal dari pemgograman linier dapat ditemukan dengan mengeksplotasi solusi dasarnya. Solusi dasar

dari pemrograman linier dalam bentuk standar dari sebanyak n-variabel dan m-batasan mengikuti ketentuan sebagai berikut:

1. Mempunyai n hingga m variabel *non basic* sama dengan nol atau Xn = 0
2. Mempunyai m variabel *basic* lebih besar atau sama dengan nol atau Xn >= 0

Ketika satu atau lebih variabel *basic* sama dengan nol, solusinya disebut *degenerate*. Solusi dasar akan sesuai dengan *feasible region*.

Langkah-langkah dalam metode simpleks terdiri dari:

- Temukan solusi dasar awal
- Jelajahi solusi dasar yang bergerak ke arah kenaikan lokal maksimum (MAX) atau penurunan (MIN) dari fungsi tujuan.
- Berhenti ketika solusi optimal ditemukan.

Software yang menyelesaikan pemrograman linier biasanya menggunakan algortima simpleks, atau mungkin algoritme simpleks yang diperbaharui, sebuah varian dari algortima simpleks asli yang diterapkan dengan lebih efisien pada komputer. Algortima lainnya digunakan untuk varian masalah pemrograman linier, seperti masalah transportasi atau pengiriman.

Pendekatan lainnya untuk menyelesaikan pemrograman linier adalah algoritme *interior point* yang dikembangkan oleh Narenda Karmarkar. Algoritme ini terbukti sangat berguna dalam masalah yang besar seperti *sparse matrix*. Berbeda dengan pendekatan simpleks, algoritma ini dimulai dari titik di dalam wilayah yang layak, dan mendekati bilangan iteratif dengan optimal.

Penyelesaian Model Pemrograman Linier: Metode Grafik

Dalam menyelesaikan permasalahan pemrograman linier, ada beberapa metode yang dapat kita gunakan, yakni pendekatan secara garfis dan pendekatan dengan cara simpleks. Pendakatan grafis biasa dipakai untuk menyelesaikan problem yang jumlah variabel keputusannya hanya dua. Sementara itu, pendekatan simpleks dikenal sebagai cara penyelesaian problem linier programking yang mempunyai jumlah varabel lebih dari dua atau bahkan bisa dipakai untuk menyelesaikan programa linier dengan variabel yang sangat besar yang tidak mungkin diselesaikan dengan pendekatan grafis.

Pada bagian ini, akan dijelaskan pendekatan penyelesaian permasalahan pemrograman linier dengan menggunakan pendekatan grafis yang dipakai dalam penyelesaian problem linier programming yang berfungsi tujuan maksimum maupun problem LP berfungsi tujuan minimum. Sementara itu, untuk penyelesaian dengan pendekatan simpleks, dibahas di bagian berikutnya dari buku ini. Target yang ingin di capai pada bagian ini, adalah kita dapat menyelesaikan permasalahan program linier dengan menggunakan metode grafik dan mempelajari problem *infeasibility, unboundedness, alternative optima,* dan *redundancy*.

Dalam program linier dengan metode grafik sering dijumpai permasalahan secara teknis, sebagai berikut:

a. *Infeasibility*, yaitu sebuah situasi yang terjadi dimana tidak terdapat daerah layak solusi yang sesuai atau memenuhi semua kendala sumber daya yang ada di sistem persamaan linier.
b. *Unboundedness*, merupakan permasalahan yang bisanya terjadi saat daerah layak yang ditemukan sangat luas dan tidak berbatas.
c. *Redundancy*, yaitu kondisi penyelesaian untuk mencari nilai optimal dari sebuah problem programa linier yang prosesnya berulang tanpa hasil perbaikan nilai optimal yang signifikan.
d. *Alternative Optimal*, yaitu merupakan situasi yang terjadi pada programa linier dimana ada solusi optimal yang berjumlah lebih dari satu titik.

Masalah-Masalah Khusus dalam Pemrograman Linier

Dalam pemrograman linier ada beberapa masalah khusus yang tidak jarang dihadapi pemrogram linier dalam proses mendapatkan keputusan yang optimal. Beberapa masalah yang sering dihadapi antara lain:

1. Solusi tidak layak (*no feasible solution*), jika tidak ada satu titikpun yang memenuhi fungsi kendala. Masalah ini dapat terjadi karena kesalahan dalam membuat formulasi pemrograman linier atau kesalahan dalam menggambarkan garis kendala, sehingga kita tidak dapat menemukan *feasible region*.

Contoh: *Maximise* $z = 5x1 + 3x2$

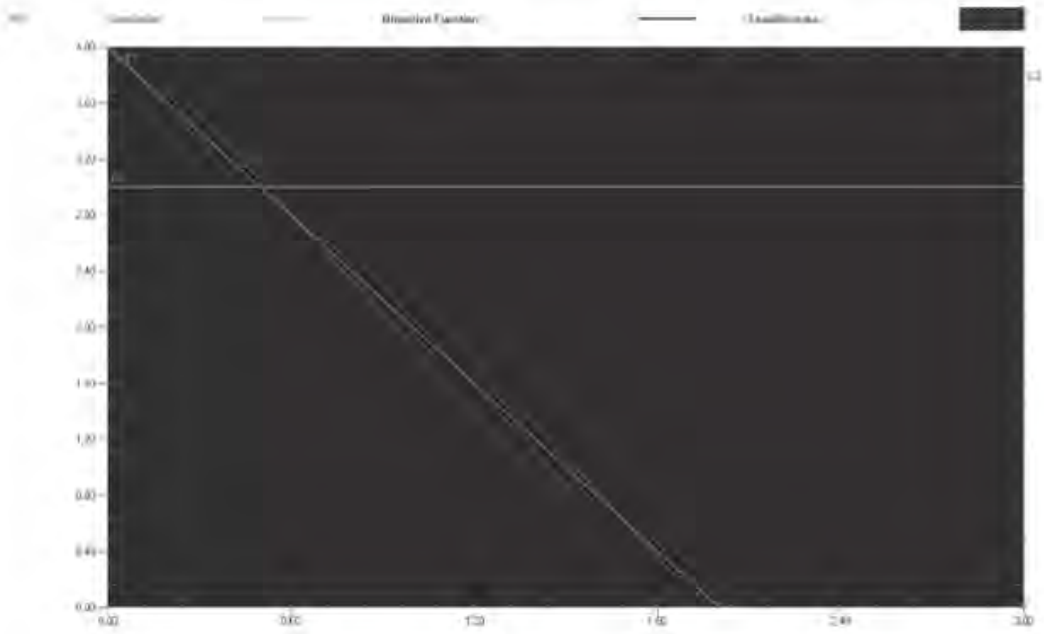
Terhadap

$4x1 + 2x2 \leq 8$

$x1 \geq 3$

$x2 \geq 3$

$x1, x2 \geq 0$



Gambar 3. 2 Solusi Tidak Layak

2. Solusi optimum lebih dari satu (*multiple optimum solution*), jika fungsi tujuan sejajar dengan fungsi kendala yang menghubungkan titik ekstrem.

Contoh: ***Max*** $z = 4x1 + 4x2$

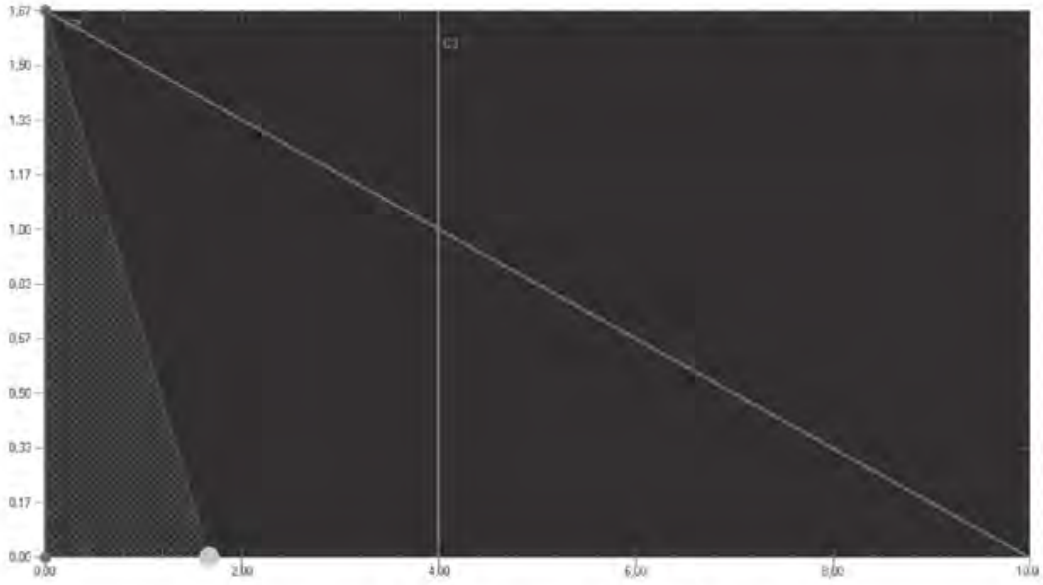
Fungsi kendala

$x1 + 6x2 \leq 10$

$6x1 + 6x2 \leq 10$

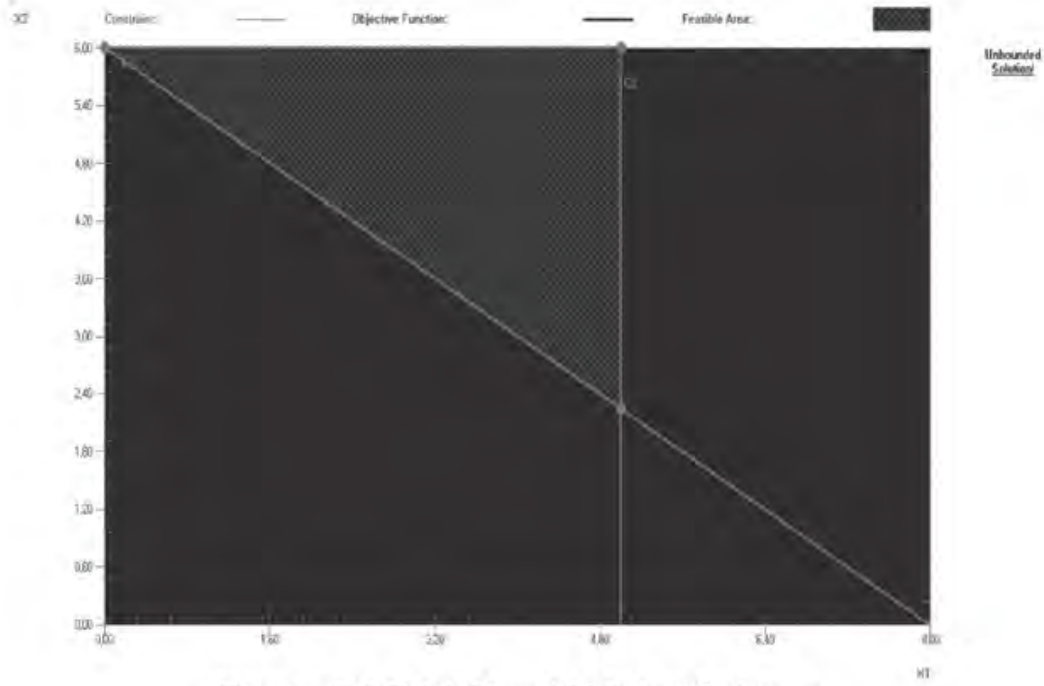
$x1 \geq 4$

$x1, x2 \geq 0$



Gambar 3. 3 Solusi Lebih dari Satu

3. Tidak memiliki solusi optimum (*un bounded solution*), jika solusi layak tidak terbentuk dan fungsi kendala tidak dapat membatasi peningkatan nilai fungsi tujuan baik kearah positif maupun negatif.



Gambar 3. 4 Tidak Memiliki Solusi Optimum

Langkah-Langkah Penyelesaian Metode Grafik

a. Gambarkan fungsi kendala dalam bentuk persamaan pada sumbu cartesius
b. Tentukan daerah solusi layak (*feasible solution*) atau area layak (*feasible region*) dengan memperhatikan tanda ketidaksamaan fungsi kendala
c. Gambarkan fungsi tujuan, geser garis tersebut ke lokasi titik solusi optimal
d. Selesaikan persamaan-persamaan pada titik solusi untuk menentukan solusi optimal
e. Untuk menentukan solusi optimal bisa memakai salah satu pendekatan yaitu pendekatan garis keuntungan (*profit line*) atau pendekatan titik sudut (*corner point*). Salah satu pendekatan tersebut di atas bisa dipakai yang akan menghasilkan nilai optimal yang sama.

Sebagai contoh kita akan menyelesaikan permasalahan pada contoh 1. Langkah paling awal untuk menyelesaikan problem programa linier dengan pendekatan garfis adalah dengan menggambar fungsi batasan sumber dayanya. Dalam menggambar fungsi batasan sumber daya atau kendala ke dalam grafik, tanda yang ada di persamaan kendala yang bentuknya masih pertidaksamaan harus diubah ke dalam bentuk persamaan agar bisa dibuat sebuah garis lurus yang memotong grafik pada sumbu x dan y, misalnya batasan sumber daya jam tenaga kerja berikut ini akan coba dibuatkan garis lurus dalam sebuah grafik:

$$30P1 + 20P2 = 2700$$

Persamaan kendala sumber daya jam tenaga kerja di atas kalau digambarkan ke dalam sebuah grafik akan memotong masing-masing sumbu x dan y. Seperti yang sudah kita pelajari dalam ilmu aljabar matematika, fungsi linier yang akan kita gambarkan ke dalam sebuah grafik merupakan sebuah garis lurus. Sehingga perlu dicari titik potong yang berupa koordinat pada kedua sumbu x dan y. Perlu diketahui bahwa sebuah garis lurus dikatakan memotong salah satu sumbu baik x ataupun y hanya jika nilai salah satu variabel sama dengan nol (0). Sehingga pada kendala yang pertama akan memotong sumbu x1 (dalam kasus di atas P1), pada saat x2 (dalam kasus ini P2) = 0, pun begitu juga kendala ini akan memotong sumbu x2, pada saat x1 = 0.

Kendala jam tenaga kerja

$$30P1 + 20P2 = 2700$$

1. Memotong sumbu P1 pada saat P2 = 0

$$30P1 + 0 = 2700$$
$$P1 = 2700/30$$
$$P1 = 90$$

2. Memotong sumbu P2 pada saat P1 = 0

$$0 + 20P2 = 2700$$
$$P2 = 2700/20$$
$$P2 = 135$$

Kendala jam tenaga kerja akan memotong sumbu P1 pada titik (90,0) dan memotong sumbu P2 pada titik (0,135)

Kendala jam kerja mesin

$$5P1 + 10P2 = 850$$

1. Memotong sumbu P1 pada saat $P2 = 0$

$$5P1 + 0 = 850$$
$$P1 = 850/5$$
$$P1 = 170$$

2. Memotong sumbu P2 pada saat P1 = 0

$$0 + 10P2 = 850$$
$$P2 = 850/10$$
$$P2 = 85$$

Kendala jam kerja mesin akan memotong sumbu P1 pada titik (170,0) dan memotong sumbu P2 pada titik (0,85)

Kendala jumlah produksi minimal

$$P1 + P2 = 95$$

1. Memotong sumbu P1 pada saat P2 = 0

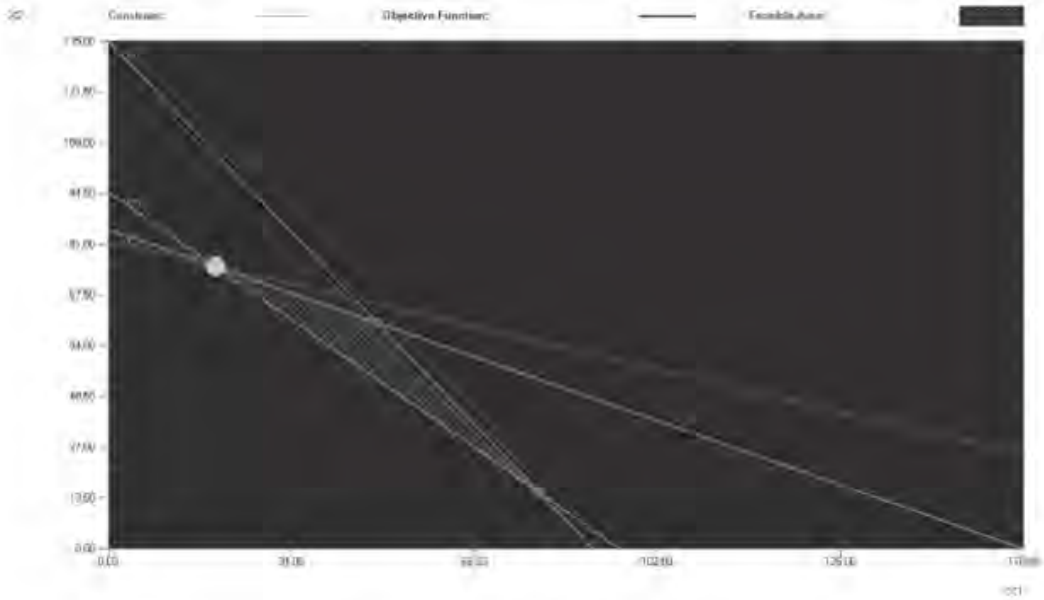
$$P1 + 0 = 95$$
$$P1 = 95$$

2. Memotong sumbu P2 pada saat P1 = 0

$$0 + P2 = 95$$
$$P2 = 95$$

Kendala jam kerja mesin akan memotong sumbu P1 pada titik (95,0) dan memotong sumbu P2 pada titik (0,95)

Berikut adalah gambar grafik dari solusi contoh 1.



Gambar 3. 5 Hasil Grafik contoh 1.

Pada gambar di atas, dapat kita lihat bahwa daerah yang diarsir adalah daerah layak atau *feasible region*. Tanda lebih kecil atau sama dengan (<=) untuk ketiga kendala/batasan diperlihatkan oleh daerah bagian sebelah kiri dari garis kendala atau batasan (daerah yang diarsir). Titik potong antara ketiga kendala di atas dapat kita cari dengan metode subtitusi atau eliminasi, sebagai contoh dapat kita lakukan yaitu antara kendala jam tenaga kerja dan kendala jam mesin,

$$30P1 + 20P2 = 2700$$

$$30P1 = 2700 - 20P2$$

(persamaan 1)

$$5P1 + 10P2 = 850$$

$$5P1 = 850 - 10P2$$

$$P1 = \frac{(850 - 10P2)}{5}$$

$$P1 = 170 - 2P2$$

(persamaan 2)

Maka, substitusikan persamaan 2 ke persamaan 1

$$30(170 - 2P2) = 2700 - 20P2$$
$$5100 - 60P2 = 2700 - 20P2$$
$$5100 - 2700 = 60P2 - 20P2$$
$$2400 = 40P2$$
$$P2 = 60$$

Maka,

$$P1 = 170 - 2P2$$
$$P1 = 170 - 2(60)$$
$$P1 = 170 - 120$$
$$P1 = 50$$

Sehingga kedua kendala akan memotong pada titik (50,60). Dari gambar 3 dapat kita lihat bahwa daerah layak berada pada titik potong antara kendala 1 dan 2 (50,60), kendala 1 dan 3 (72,22,26,67) dan kendala 2 dan 3 yaitu (20,75).

Sekarang kita akan menyelesaikan permasalahan di atas menggunakan pendekatan titk pojok/sudut (*corner point*), yang berarti kita harus menenetukan nilai tertnggi dari beberapa titik yang masuk dalam daerah layak (*feasible region*). Dengan menggunakan fungsi tujuan:

$$MAX\ z = 20P1 + 60P2$$

Maka keuntungan pada titik:

- (50,60) adalah (20*50) + (60*60) = 4600
- (72,22,26,67) adalah (20*72,27) + (60*26,67) = 3045,6
- **(20,75) adalah (20*20) + (60*75) = 4900 (tertinggi)**

Dari hasil di atas, seperti diketahui bahwa kasus tersebut adalah problem dengan fungsi tujuan maksimal, maka yang dicari adalah nilai tertinggi. Apabila diasumsikan yang dicari adalah maksimal keuntungan, maka keuntungan paling besar/tinggi berada di titik (20,75), sehingga rekomendasi yang diberikan ke perusahaan adalah memproduksi produk 1 sebesar 20 ton dan produk 2 sejumlah 75 ton, dan perusahaan akan memperoleh keuntungan optimal sebesar 4900.

Penyelesaian Pemrograman Linier dengan R

Tersedia beberapa cara untuk menyelesaikan model pemrograman linier. Daftar dari *solver* tersebut bisa dicek dan diunduh di tautan: http://bit.ly/1zkJpVw. Beberapa dari *solver* tersebut dapat digabungkan ke dalam program yang lebih besar untuk mengembangkan masalah optimasi. Beberapa diantaranya ditulis menggunakan bahasa C dan juga diterapkan di *pakcages* R. *Packages* berikut dapat menjadi pilihan bagi pengguna R.

- lp_solve adalah penerapan melalui *packages* **lpSolve** dan **lpSolveAPI**
- GLPK merupakan penerapakn melalui *package* **Rglpk**
- SYMPHONY adalah implementasi dari **Rsymphony**

Seluruh *solver* tersebut diimplementasikan melalui fungsi-fungsi di R. *Solver* ini juga dapat dimasukkan ke dalam program yang lebih besar. Beberapa dari *package* tersebut mempunyai fungsi-fungsi yang dapat menerjemahkan program ILP dan MILP dari file, dan menuliskan itu ke dalam bentuk standar seperti CPLEX, MPS atau AMPL/MathProg.

Kebanyakan dari *package* di R menyelesaikan pemrograman linier dengan menerapkan *solver* sebagai fungsi, dimana variabel inputnya adalah:

- Sebuah variabel karakter menunjukkan jika kita akan menyelesaikan masalah maksimasi atau minimasi
- Beberapa vektor dengan koefisien biaya c dan nilai sisi kanan b
- Sebuah matriks dengan koefisien A
- Sebuah vektor karakter dengan tanda untuk kendala.

Untuk model ILP dan MILP, sebuah vektor tambahan yang menunjukkan variabel mana yang merupakan bilangan bulat. Dengan alternatif beberapa variabel logika yang menunjukkan jika semua variabel integer atau biner.

Pemrograman Linier menggunakan *package* lpSolve

Dalam beberapa masalah sederhana, seperti yang ditampilkan pada contoh 1, mendefinisikan parameter yang digunakan akan mudah, jika kita mengetahui tentang notasi di R. yang perlu diingat adalah R tidak menyediakan *Package* lpSolve secara default, sehingga kita harus mendownloadnya terlebih dahulu di:"https://cran.r-project.org/web/

packages/lpSolve/index.html", untuk manual penggunaan dari *package* lpSolve dapat didownload di: "https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf" dan juga tersedia di lampiran buku ini. Untuk memanggil fungsi dari *package* lpSolve, dapat menggunakan kode berikut.

```
library(lpSolve)
```

Seperti yang sudah dibahas sebelumnya di Bab 1, bahwa *package* dalam bahasa R sebenarnya adalah kumpulan fungsi dari program untuk menyelesaikan masalah tertentu. Sehingga dalam *package* lpSolve juga terdiri dari sekumpulan *script* program, yang dapat kita lihat dengan mengetikkan lp di R Console.

```
"> lp
function (direction = "min", objective.in, const.mat, const.dir,
const.rhs, transpose.constraints = TRUE, int.vec, presolve = 0,
compute.sens = 0, binary.vec, all.int = FALSE, all.bin = FALSE,
scale = 196, dense.const, num.bin.solns = 1, use.rw = FALSE)
{
    if (direction == "min")
            direction <- 0
    else if (direction == "max")
            direction <- 1
    else stop("Direction must be 'max' or 'min'")
    if (is.data.frame(objective.in)) {
            if (ncol(objective.in) > 1)
                    stop("Objective vector has more than one column")
            objective.in <- unlist(objective.in)
            names(objective.in) <- NULL
    }
    objective <- c(0, objective.in)
    solution <- numeric(length(objective.in))
    status <- objval <- 0
    x.count <- length(objective.in)
    constraints <- 0
    const.count <- 0
```

```
use.dense <- FALSE
if (!missing(const.mat)) {
   dense.const <- 0
   dense.const.nrow = 0
   dense.ctr = 0
   if (is.data.frame(const.mat)) {
      cm <- as.numeric(unlist(const.mat))
      names(cm) <- NULL
      const.mat <- matrix(cm, nrow = nrow(const.mat))
   }
   const.mat[is.na(const.mat)] <- 0
   if (transpose.constraints)
      const.mat <- t(const.mat)
   const.count <- ncol(const.mat)
}
else {
   if (missing(dense.const)) {
      dense.const <- 0
      dense.const.nrow = 0
      dense.ctr = 0
   }
   else {
      if (!is.matrix(dense.const))
         stop("Dense constraints need to be matrix or data frame.")
      if (is.data.frame(dense.const))
         dense.const <- as.matrix(dense.const)
      if (ncol(dense.const) != 3)
         stop("Dense constraints must be in three columns.")
      dimnames(dense.const) <- list(NULL, c("Const", "Var",
         "Value"))
      dense.const <- dense.const[order(dense.const[, "Const"]),
         , drop = FALSE]
```

```
        const.indices <- unique(dense.const[, "Const"])
        if (length(const.indices) != max(const.indices))
          stop("Error in constraint numbering")
        const.count <- max(const.indices)
        dense.ctr <- table(dense.const[, "Const"])
        names(dense.ctr) <- NULL
        dense.const <- dense.const[, c("Var", "Value"), drop = FALSE]
        dense.const.nrow <- nrow(dense.const)
        use.dense <- TRUE
    }
}
const.dir.num <- rep(-1, length(const.dir))
const.dir.num[const.dir == "<" | const.dir == "<="] <- 1
const.dir.num[const.dir == "=" | const.dir == "=="] <- 3
const.dir.num[const.dir == ">" | const.dir == ">="] <- 2
if (any(const.dir.num == -1))
    stop("Unknown constraint direction found \ \n")
if (is.data.frame(const.rhs))
    const.rhs <- as.matrix(const.rhs)
const.rhs <- c(const.rhs)
names(const.rhs) <- NULL
if (!missing(const.mat)) {
    big.const.mat <- rbind(const.mat, const.dir.num, const.rhs)
    constraints <- c(0, c(big.const.mat))
}
else if (!missing(dense.const)) {
    dense.ctr <- rbind(dense.ctr, const.dir.num, const.rhs)
    big.const.mat <- 0
    constraints <- 0
}
if (!missing(all.int) && all.int) {
    int.vec <- 1:length(solution)
```

```
    int.count <- length(int.vec)
  }
  else {
    if (missing(int.vec)) {
      int.count <- 0
      int.vec <- 0
    }
    else int.count <- length(int.vec)
  }
  if (!missing(all.bin) && all.bin) {
    binary.vec <- 1:length(solution)
    bin.count <- length(binary.vec)
  }
  else {
    if (missing(binary.vec)) {
      bin.count <- 0
      binary.vec <- 0
    }
    else bin.count <- length(binary.vec)
  }
  if (length(binary.vec) == length(solution))
    all.bin <- TRUE
  if (num.bin.solns > 1)
    if (all.bin)
      solution <- c(0, rep(solution, num.bin.solns))
    else {
      warning("Num.bin.solns can only be > 1 if all variables are
binary")
      num.bin.solns <- 1
    }
  sens.coef.from <- sens.coef.to <- 0
  duals <- duals.from <- duals.to <- 0
```

```
    if (compute.sens != 0) {
      sens.coef.from <- sens.coef.to <- numeric(x.count)
      duals <- duals.from <- duals.to <- numeric(x.count +
        const.count)
    }
    if (num.bin.solns > 1 && use.rw == TRUE)
      tmp <- tempfile()
    else tmp <- "Nobody will ever look at this"
    if (missing(dense.const) || !is.matrix(dense.const)) {
      dense.col <- dense.val <- 0
    }
    else {
      dense.col = dense.const[, "Var"]
      dense.val = dense.const[, "Value"]
    }
    lp.out <- .C("lpslink", direction = as.integer(direction),
      x.count = as.integer(x.count), objective = as.double(objective),
      const.count = as.integer(const.count), constraints =
as.double(constraints),
      int.count = as.integer(int.count), int.vec = as.integer(int.vec),
      bin.count = as.integer(bin.count), binary.vec = as.integer(binary.
vec),
      num.bin.solns=as.integer(num.bin.solns), objval=as.double(objval),
      solution = as.double(solution), presolve = as.integer(presolve),
      compute.sens = as.integer(compute.sens), sens.coef.from =
as.double(sens.coef.from),
      sens.coef.to = as.double(sens.coef.to), duals = as.double(duals),
      duals.from = as.double(duals.from), duals.to = as.double(duals.to),
      scale = as.integer(scale), use.dense = as.integer(use.dense),
      dense.col = as.integer(dense.col), dense.val = as.double(dense.val),
      dense.const.nrow = as.integer(dense.const.nrow), dense.ctr =
as.double(dense.ctr),
```

```
        use.rw = as.integer(use.rw), tmp = as.character(tmp),
        status = as.integer(status), PACKAGE = "lpSolve")
    lp.out$objective <- objective.in
    lp.out$constraints <- big.const.mat
    if (any(names(version) == "language"))
        class(lp.out) <- "lp"
    else oldClass(lp.out) <- "lp"
    return(lp.out)"
}
<bytecode: 0x074331d4>
<environment: namespace:lpSolve>
```

Berikutnya, kita akan menyelesaikan masalah pada contoh 1 menggunakan R. Kode berikut akan digunakan untuk menyelesaikan pemrograman linier dengan 2 variabel, diketik di R editor.

```
library(lpSolve)

#defining parameters
obj.fun <- c(20, 60)
constr <- matrix(c(30, 20, 5, 10, 1, 1), ncol = 2, byrow= TRUE)
constr.dir <- c("<=", "<=", ">=")
rhs <- c(2700, 850, 95)

#solving model
prod.sol <- lp("max", obj.fun, constr, constr.dir, rhs, compute.sens
= TRUE)

#accessing to R output
prod.sol #objective function value
prod.sol$solution #decision variables values
prod.sol$duals #includes duals of constraints and reduced costs of
variables
```

```
#sensibility analysis results
prod.sol$duals.from
prod.sol$duals.to
prod.sol$sens.coef.from
prod.sol$sens.coef.to
```

Pada bagian *#defining parameters* kita akan mendefinisikan fungsi tujuan dan fungsi kendala.

Fungsi tujuan pada kasus ini adalah $MAX\ z = 20P1 + 60P2$, dinyatakan dalam kode R sebagai berikut

```
obj.fun <- c(20, 60)
```

fungsi tujuan ini didefinisikan dalam sebuah vektor bernama obj.fun yang berisi angka 20 dan 60. Pada kasus ini berarti variabel P1 bernilai 20 dan P2 bernilai 60.

Selanjutnya untuk fungsi kendala, dinyatakan dalam kode R sebagai berikut

```
constr <- matrix(c(30, 20, 5, 10, 1, 1), ncol = 2, byrow= TRUE)
```

matriks cosntr yang berisi nilai kendala didefinisikan dengan jumlah kolom sebanyak 2 (ncol = 2), yang akan membuat matriks ini berdimensi 2x3 (karena diisi perbaris menggunakan byrow = TRUE). Sehingga tampilan matriks kendala adalah sebagai berikut.

```
     [,1] [,2]
[1,]   30   20
[2,]    5   10
[3,]    1    1
```

```
constr.dir <- c("<=", "<=", ">=")
```

menunjukkan tanda pertidaksamaan untuk masing-masing fungsi kendala. Dinyatakan dalam bentuk vektor yang disimpan dalam variabel constr.dir

```
rhs <- c(2700, 850, 95)
```

menyatakan vektor yang bernama rhs, berisi nilai sisi kanan dari masing-masing fungsi kendala.

Selanjutnya, bagian *#solving model*, seluruh parameter yang akan telah didefinisikan akan dipanggil dengan fungsi lp(), dan hasil dari masalah ini disimpan dalam sebuah variabel bernama prod.sol.

```
prod.sol <- lp("max", obj.fun, constr, constr.dir, rhs, compute.sens = TRUE)
```

pada kode R di atas, dapat dilihat bahwa ada tulisan "max" yang menyatakan bahwa model ini bertujuan untuk memaksimalkan. Ketika model yang diinginkan bertujuan untuk meminimalkan, maka gunakan "min"

Bagian #accessing to R output akan menampilkan ouput dari model pemrograman linier.

Nilai dari fungsi tujuan dinyatakan dengan kode

```
prod.sol #objective function value
> prod.sol #objective function value
Success: the objective function is 4900
```

Dapat dilihat bahwa, model ini mempunyai solusi, dinyatakan dengan kata Success, sedangkan nilai dari fungsi tujuannya adalah 4900. Nilai dari variabel keputusan dapat dilihat pada

```
> prod.sol$solution #decision variables values
[1] 20 75
```

Ini berarti bahwa variabel keputusan untuk P1 bernilai 20 dan P2 bernilai 75. Sedangkan jika ingin melihat dual nya dapat menggunakan kode

```
> prod.sol$duals #includes duals of constraints and reduced costs of variables
[1]  0  8 -20  0  0
```

Dan jika ingin mengetahui analisis sensitifitas dari model ini, dapat dilihat pada bagian *#sensibility analysis results*

```
> #sensibility analysis results
> prod.sol$duals.from
[1] -1.0e+30  5.5e+02  8.5e+01 -1.0e+30 -1.0e+30
> prod.sol$duals.to
[1] 1.0e+30 9.5e+02 1.1e+02 1.0e+30 1.0e+30
```

```
> prod.sol$sens.coef.from
[1] -1e+30  4e+01
> prod.sol$sens.coef.to
[1] 3e+01 1e+30
```

Penyelesaian model pemrograman linier yang lebih kompleks, kita akan bahas di Bab 7 dalam buku ini.

Untuk masalah yang lebih kompleks, ada cara yang lebih efisien yaitu dengan melewati parameter model daripada melist semua variabel. Ini adalah kasus yang sudah didefinisikan di bagian contoh 2, sebuah contoh sederhana dari masalah transportasi. Kode berikut mendefinisikan matriks A untuk seluruh angka dari pabrik m dan tujuan n. dari masalah transportasi.

```
library(lpSolve)

"#defining parameters
#origins run i in 1:m
#destinations run j in 1:n

obj.fun <- c(8, 6, 3, 2, 4, 9)
m <- 2
n <- 3

constr <- matrix(0, n+m, n*m)
for(i in 1:m)
    {for(j in 1:n){
     constr[i, n*(i-1) + j] <- 1
    constr[m+j, n*(i-1) + j] <- 1 } }

constr.dir <- c(rep("<=", m), rep(">=", n))

rhs <- c(70, 40, 40, 35, 25)
```

```
#solving LP model
prod.trans <- lp("min", obj.fun, constr, constr.dir, rhs, compute.sens
= TRUE)

#LP solution
prod.trans$obj.val
sol <- matrix(prod.trans$solution, m, n, byrow=TRUE)
sol
prod.trans$duals

#sensitivity analysis of LP
prod.trans$duals.from
prod.trans$duals.to
prod.trans$sens.coef.from
prod.trans$sens.coef.to
```

Kode di atas akan menghasilkan penyelesaian sebagai berikut.

```
> prod.trans
Success: the objective function is 365
> sol <- matrix(prod.trans$solution, m, n, byrow=TRUE)
> sol
     [,1] [,2] [,3]
[1,]    0   35   25
[2,]   40    0    0
> prod.trans$duals
 [1]  0 -2  4  6  3  4  0  0  0  0  8
>
> #sensitivity analysis of LP
> prod.trans$duals.from
 [1] -1.000000e+30   4.000000e+01   5.000000e+00   0.000000e+00
-3.552714e-15 0.000000e+00 -1.000000e+30 -1.000000e+30 -1.000000e+30
-1.000000e+30 -3.500000e+01
```

```
> prod.trans$duals.to
 [1] 1.0e+30 7.5e+01 4.0e+01 4.5e+01 3.5e+01 3.5e+01 1.0e+30 1.0e+30
1.0e+30 1.0e+30 0.0e+00
> prod.trans$sens.coef.from
[1]  4.000000e+00   4.000000e+00   0.000000e+00  -2.000000e+00
2.220446e-15  1.000000e+00
> prod.trans$sens.coef.to
[1] 1.0e+30 1.0e+01 1.1e+01 6.0e+00 1.0e+30 1.0e+30"
```

Penjelasan mengenai masalah transportasi dan cara penyelesaiannya akan dibahas lebih lanjut di Bab selanjutnya.

Analisis Sensitifitas Secara Grafis

Analisa sensitivitas dipergunakan untuk melihat besarnya pengaruh perubahan beberapa sumber daya terhadap penylesaian (solusi) optimal yang sudah didapatkan. Misalnya, pada perubahan kapasitas penyediaan jumlah tenaga kerja atau perubahan kemampuan penyediaan bahan baku yang akan mempengaruhi keputusan optimal yang sudah dilakukan sebelumnya. Selain itu perubahan yang mungkin terjadi adalah menyangkut harga atau keuntungan yang kecenderungan perubahannya sangat dinamis yang tentu saja akan mempengaruhi hasil optimal yang sudah didapatkan. Sehingga pertanyaan yang mestinya akan menjadi pertanyan pengambil keputusan saat perubahan data tersebut terjadi adalah apakah solusi optimal yang sudah didapatkan sebelumnya masih berlaku dengan adanya perubahan-perubahan tersebut? Analisa sensitivitas akan mencoba membantu memberikan jawaban pertanyaan diatas dan juga pertanyaan lebih jauh misalnya "jika dilakukan perubahan sumber daya, maka seberapa jauh/besar perubahan yang diperkenankan tanpa mengubah soluasi optimal terlalu jauh? Juga pertanyaan terkait dengan prioritas mana yang perlu dilakukan perubahan lebih dulu jika terdapat lebih dari satu sumber daya yang bisa diubah.

Ada 3 pertanyaan yang bisa dijawab saat memutuskan untuk melakukan analisa sensitivitas, antara lain:

1. Sumber daya atau batasasan yang mana yang bisa dilakukan perubahan dan jika bisa dilakukan perubahan, seberapa besar dapat dinaikkan atau diturunkan dengan harapan bisa menaikkan nilai fungsi tujuan (Z).

2. Jika diputuskan dilakukan perubahan sumber daya atau batasan dan terdapat lebih dari satu batasan, sumber daya yang mana yang didahulukan untuk dilakukan perubahan?
3. Perubahan yang bisa dilakukan juga bisa dilakukan pada koefisien fungsi tujuan. Sehingga seberapa besar koeffisien fungsi tujuan bisa dinaik-turunkan tanpa banyak merubah hasil optimal.

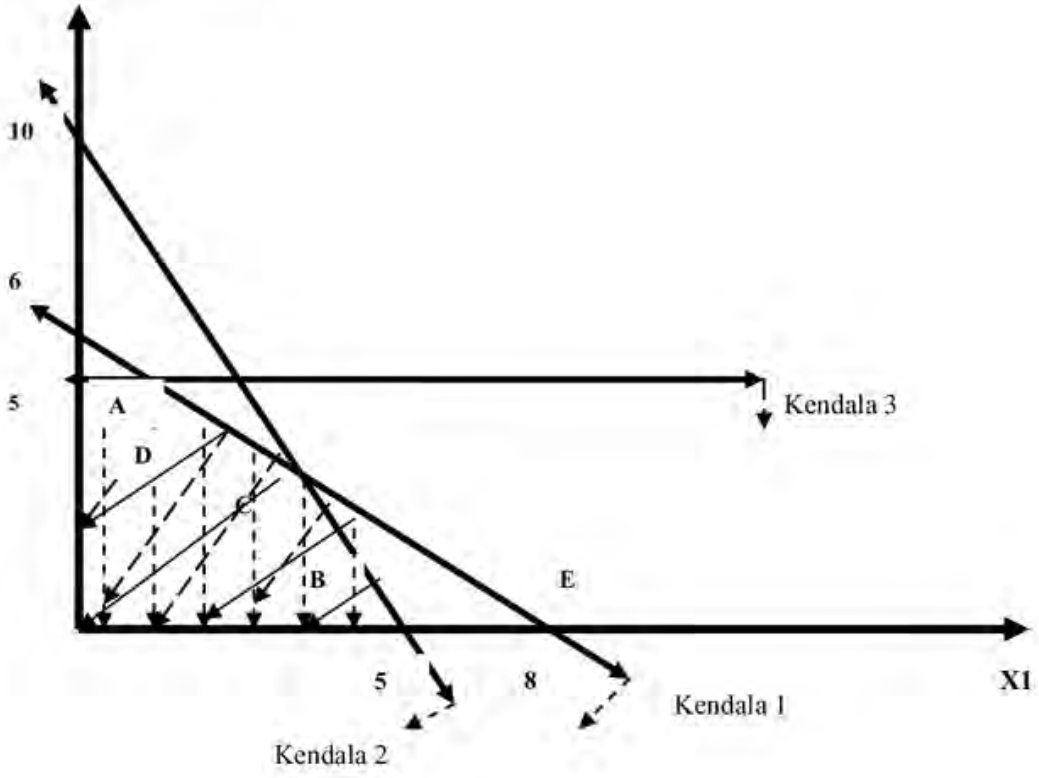
Contoh dan penyelesaian soal di bawah ini merupakan contoh soal kombinasi produksi yang diambil dari soal di buku Yamit (1993) terkait linier programming dan analisa sensitivitas: :

Selesaikan masalah production mix produk di bawah ini:

Z maks = 40X1 + 20X2

Kendala:

1. $3X1 + 4X2 \leq 24$ (bahan baku A)
2. $2x1 + X2 \leq 10$ (jam Tenaga Kerja)
3. $X2 \leq 5$ (bahan baku B)
4. Nonnegativity



Pada grafik tersebut di atas dapat kita cari solusi optimalnya yang berada pada titik B (5,0) dimana X1 = 5 unit dan X2 = 0 unit dengan nilai z = 200. Dari perhitungan pencarian solusi optimum dan grafik di atas bisa diidentifikasi 2 macam kendala yaitu:

- Kendala yang bersifat habis terpakai atau *Scarce* (*full capacity*)
- Kendala *redundant* (*idle capacity*) atau kendala yang berlebihan

Dari titik optimal yaitu B dapat dilihat bahwa pada titik optimum dipotong oleh kendala 2, hal ini menunjukkan bahwa kendala 2 (jam tenaga kerja) dalam keadaan *full capacity* (habis terpakai atau *scarce*), untuk membuktikannya dilakukan pengecekan :

Kendala [2] : $2X1 + X2 \leq 10$ dimana 10 merupakan jam tenaga kerja yang tersedia.

Yang dipakai = 2(5) + 0 = 10, jadi jam tenaga kerja yang tersedia = jam tenaga kerja yang terpakai. Berarti kendala jam tenaga kerja *full capacity*.

Persoalan selanjutnya adalah seberapa besar jam tenaga kerja dapat dinaikkan sehingga memberikan kenaikan nilai Z. Berikutnya kita coba melakukan perubahan kapasitas jam tenaga kerja B.

Perubahan Kapasistas Sumber Daya

[1] perubahan jam tenaga kerja

Dengan melihat grafik di atas, bisa disimpulkan bahwa kemampuan penyediaan jam tenaga kerja sebenarnya bisa di*upgrade* sampai mencapai titik E, dimana titik E berada pada koordinat kartesisus (8,0) atau X1 = 8 dan X2 = 0. Nilai koordinat x dan y di titik E tersebut berikutnya bisa disubtitusikan pada kendala kendala [2]: 2x1 + X2 = 2(8) + 0 = 16. Ini berarti bisa dikatakan bahwa jumlah maksimum jam tenaga kerja sebenarnya dapat dinaikkan sebesar 16 jam – 10 jam = 6 jam, sehingga daerah feasible berubah menjadi A-D-E dengan solusi optimum tercapai pada titik E. Aapabila diputuskan jam tenaga kerja ditambah sebesar itu, maka bisa dipastikan akan menaikkan nilai fungsi tujuan yang baru (Z) menjadi sebesar Z = 40 (8) + 20 (0) = 320 yang berarti perubahan tersebut akan mengakibatkan kenaikan Z sebesar Rp.120,

[2] Perubahan bahan baku A

Pada kendala bahan baku A terdapat sifat kendala yang versifat berlebihan berlebihan (*redundancy*). Sifat berlebihan ini bisa dibuktikan dengan meng-input nilai optimum B (5,0) pada kendala [1] : 3(5) + 4(0) = 15 kg. Ini berarti bahan baku A yang terpakai untuk proses produksi optimal sebenarnya hanya 15 kg,s edangkan ketersediaan awal bahan baku A disediakan sebesar 24 kg. Sehingga bisa disimpulkan bahwa bahan baku A sebaiknya diturunkan maksimum sesuai dengan kebutuhan riil-nya yakni 15 kg, atau turun sebesar 24 – 15 = 9 kg, dimana penurunan ini tidak mempengaruhi nilai fungsi tujuan Z.

[3] Perubahan Bahan Baku B

Pada kendala bahan baku B, kendala juga bersifat berlebihan hal ini dapat dibuktikan dengan memasukkan nilai optimum B (5,0) pada kendala [3]. Pada kendala [3] bahan baku B tidak digunakan untuk memproduksi X2 sehingga sebenarnya bahan baku B yang dibutuhkan sebesar 0 kg, sementara sediaan bahan baku B sebesar 5 kg. Oleh karena itu persediaan bahan baku B dapat diturunkan sampai dengan 0 kg atau turun sebesar 5 kg yang tidak akan mempengaruhi nilai fungsi tujuan Z.

Prioritas Perubahan Sumber Daya

Dari data perubahan-perubahan sumber daya yang tersedia dan bisa dilakukan sebelumnya, kita tidak tahu sumber daya mana yang harus diprioritaskan untuk dilakukan perubahan lebih dulu. Sehingga diperlukan sebuah parameter tertentu yang dipakai sebagai alat ukur untuk mengambil keputusan pemilihan sumber daya mana yang bisa diubah lebih dulu. Salah satu parameter yang bisa dipakai untuk memutuskan sumber daya yang diprioritaskan untuk dilakukan perubahan adalah dengan menghitung nilai sumbangan per unit, yang oleh Yamit (1993) Di antara perubahan-perubahan sumberdaya di atas, kendala atau sumberdaya mana yang diprioritaskan untuk ditambah tergantung dari nilai sumbangan per unit, yang dirumuskan (Yamit, 1993):

$$Yi = \frac{\text{Perubahan maksimum nilai Z optimal}}{\text{Perubahan maksimum dalam sumber daya i}}$$

Hasil perhitungan perubahan sumberdaya dapat dilihat pada tabel berikut:

Kendala	Kolom (1) Tipe Kendala	Kolom (2) Maksimum Penambahan	Kolom (3) Maksimum Penambahan Z	Yi (Rp/unit) Kolom3 / kolom 2
[1] Bahan Baku A	Berlebih	15 -24 = - 9	200 – 200 = 0	0
[2] Jam tenaga kerja	Scarce	16 – 10 = 6	320 – 200 = 120	20
[3] bBahan baku B	Berlebih	0 – 5 = -5	200 – 200 = 0	0

Sumber:(Yamit, 1993)

Dari hasil perhitungan tersebut menunjukkan bahwa kendala [2] (jam tenaga kerja) diprioritaskan untuk ditambah karena memberikan nilai sumbangan per unit yang lebih besar.

Perubahan Koefisien Fungsi Tujuan

Sebelumnya sudah dijelaskan perubahan sumberdaya dan pengaruhnya terhadap nilai Z, selanjutnya seberapa besar koefisien fungsi tujuan dapat dinaikkan atau diturunkan tanpa mengubah solusi optimum?

Dari contoh di atas diketahui bahwa:

fungsi tujuan: Zmks = 40X1 + 20X2.

Jika Koefisien X1 disimbolkan B1 dan koefisien X2 adalah B2 maka fungsi tujuan diatas akan menjadi :

Z = B1X1 + B2X2

Kasus 1 ; Perubahan koefisien B1

Seberapa besar B1 dapat dinaik-turunkan (dengan asumsi bahwa B2 tetap) sehingga Titik B tetap optimal? Naik-turun nilai B1 dapat ditentukan dengan menyamakan slope Z dengan slope kendala [2] .

Jika fungsi tujuan Z = B1X1 + 20X2, maka slope Z=B1/20.

Slope kendala [2] = 2/1 maka nilai B1 bisa berubah:

B1/20 = 2/1 atau B1 = 40---------→ hal ini menunjukkan bahwa koefisien fungsi tujuan akan berubah nilai optimalnya jika dinaik-turunkan

Kasus 2; Perubahan koefisien B2

Jika fungsi tujuan semula Z = 40X1 + 30X2 maka jika B1 tetap dan B2 dinaikturunkan, maka fungsi tujuan akan berubah:

Z = 40X1 + B2X2. Slope fungsi tujuan: Z = 40/B2 dan slope kendala [2] : 2/1

Sehingga 40/B2 = 2/1 atau B2 = 20 atau agar tetap optimal nilai B2 minimal/maksimal bisa diturunkan sampai dengan 20.

Latihan: selesaikan formulasi linier berikut ini dan coba analisa sensitivitas-lah perubahan apa yang diperbolehkan dan pengaruhnya terhadap nilai optimal:

Zmaks = 40X1 + 30X2

d.k [1] 2x1 + 3X2 ≤ 60

[2] 2X2 ≤ 30

[3] 2X1 + X2 ≤ 40

[4] non negative

Pemrograman Linier Dengan Metode Simpleks

Metode simpleks merupakan salah satu pendekatan programa linier dalam bentuk khusus yang digunakan untuk memecahkan masalah atau problem LP yang dengan jumlah variabel labih dari dua, yang tidak bisa diselesaikan dengan pendekatan grafis. Pada dasanya, proses pencarian solusi optimal dari metode simpleks ini menggunakan teknik eliminasi dari Gauss Jordan. Pada pendekatan simpleks, proses pencarian nilai optimal dilakukan dengan mengecek dan memeriksa titik paling luar atau titik ekstrim *step by step* dan satu per satu dengan menggunakan cara *iterative* atau berulang. Jadi dengan kata lain, penentuan *optial solution* pada pendekatan simpleks yang dilakukan bertahap disebut sebagai *iterative* (iterasi). Karena dilakukan bertahap, maka hasil dari setiap langkah atau iterasi yang dilakukan sangat tergantung iterasi sebelumnya atau dengan kata lain nilai dan keakuratan hasil perhitungan iterasi ke i akan sangat bergantung pada nilai dan keakuratan perhitungan yang dilakukan pada iterasi sebelumnya (i-1). Beberapa istilah atau sebutan khusus yang biasa dipergunakan dalam pendekatan simpleks yang akan sering kita sebut dan pakai, misalnya:

1. Iterasi merupakan tahap per tahap pencarian solusi optimal dalam metode simpleks dimana tahap perhitungan yang dilakukan tersebut akan sangat berpengaruh terhadap perhitungan setelahnya.

2. Variabel non basis di dalam pendekatan simpleks merupakan variabel yang besarannya dikondisikan bernilai nol (0) pada semua iterasi. Jumlah variabel basisi di dalam pendekatan simpleks selalu dianlogkan sama dengan jumlah variabel yang terlibat dalam sistem persamaan programa linier.
3. Variabel basis di metode simpleks adalah variabel yang bernilai selain nol (0) pada sembarang iterasi. Pada iterasi pertama, variabel basis diberi nama variabel *slack* (apabila ditemukan persamaan kendala berbentuk lebih kecil sama dengan atau ≤) atau dinamakan variabel *artificial* (buatan) (jika ditemukan pada persamaan kendala berbentuk lebih dari sama dengan atau persamaan kendala berbentuk sama dengan / ≥ atau =). Untuk jumlah variabel basis (variabel *slack* atau variabel *artificial*) yang perlu ditambahkan ke tiap persamaan kendala sangat tergantung dari jumlah persamaan kendala. Setiap persamaan kendala yang akan memiliki satu variabel basis baik itu variabel *slack* atau variabel buatan.
4. Nilai ruas kanan kendala dari masing-masing kendala merupakan nilai sumber daya yang saat ini tersedia. Pada iterasi pertama, nilai kanan tabel simpleks sama dengan nilai sumber daya yang tersedia karena masih belum ada aktivitas/iterasi yang dilakukan.
5. Variabel *slack* di pendekatan simpleks merupakan variabel yang ditambahkan ke dalam persamaan matematis kendala yang berfungsi merubah sistem pertidaksaamaan persamaan kendala menjadi bentuk persamaan agar bisa dikerjakan dengan pendekatan simpleks. Dengan kata lain, variabel *slack* diperlukan untuk merubah dan mengkonversikan tanda matematis (≤) menjadi tanda matematis (=). Proses penambahan variabel *slack* ini dilakukan di proses paling awal yang disebut tahap inisialisasi atau tahap pengkondisian sistem persamaan. Perlu diingat bahwa, variabel *slack* yang sudah ditambahkan di setiap persaman kendala akan berfungsi sebagai variabel basis pada iterasi-iterasi berikutnya di metode simpleks.
6. Variabel surplus dalam metode simpleks dikenal sebagai variabel pengurang ruas kiri persamaan kendala. Untuk persamaan kendala yang berbentuk pertidaksamaan (≥) mengindikasikan bahwa ruas kiri kendala perlu dikurangi dengan 'sesuatu" agar menjadi bentuk yang bisa dikerjakan dengan metode simpleks yakni bentuk persamaan (=). Proses penambahan variabel *slack* ini dilakukan diproses paling

awal yang disebut tahap inisialisasi atau tahap pengkondisian sistem persamaan.

7. Variabel buatan atau artificial di dalam metode simpleks merupakan variabel yang ditambahkan pada persamaan kendala yang berbentuk ≥ atau =. Oleh karena persamaan kendala yang berbentuk ≥ atau = tidak punya variabel basis maka proses penambahan tersebut berfungsi sebagai variabel basis awal yang dipakai di iterasi pertama. Proses penambahan variabel *slack* ini juga dilakukan diproses paling awal yang disebut tahap inisialisasi atau tahap pengkondisian sistem persamaan. Variabel buatan ini pada akhir iterasi metode simpleks harus bernilai nol (0) atau hilang dari tabel simpleks dengan alasan bahwa variabel ini sebenarnya tidak eksis atau secara riil tidak ada, sehingga harus dinetralkan atau dihilangkan atau di-nol-kan di akhir iterasi.
8. Kolom kunci atau kolom pivot akan sering disebut di metode simpleks yang akan memuat variabel masuk (*incoming variable*). Perlu diperhatikan bahwa nilai koefisien dari kolom kunci ini akan dipakai sebagai nilai pembagi nilai index atau nilai ruas kanan tabel simpkes untuk menentukan baris kunci atau baris pivot.
9. Baris pivot atau baris kunci dalam proses tabel simpleks akan dipakai acuan sebagai baris yang memuat variabel keluar (*outcoming variable*).
10. Nilai kunci atau biasa disebut sebagai elemen pivot dalam metode simpleks berada di posisi perpotongan antara kolom kunci dan baris kunci. Nilai atau elemen kunci (pivot) ini merupakan nilai penting yang akan dipakai sebagai dasar perhitungan untuk iterasi simpleks berikutnya.
11. Variabel masuk atau *incoming variable* dari sebuah iterasi di metode simpleks merupakan variabel yang akan menjadi variabel basis pada iterasi selanjutnya. Variabel masuk ini dipilih dari variabel non basis pada setiap iterasi di proses simpleks. Variabel masuk yang notabene berupa variabel non basis yang akan masuk di kolom tempat variabel variabel basis pada iterasi berikutnya akan bernilai positif (+).
12. Variabel keluar atau *outcoming variable* dalam metode simpleks merupakan variabel yang basis yang berda di kolom variabel basis yang pada iterasi berikutnya akan dikeluarkan dan diganti variabel masuk (*incoming variable*) yang merupakan variabel non basis. Variabel keluar ini pada iterasi berikutnya akan bernilai nol.

Perlu ingat lagi bahwa sebelum menentukan solusi optimal, perhitungan iteratif yang berdasarkan iterasi-iterasi, perlu dilakukan pengkondisian bentuk persamaan kendala menjadi bentuk umum dan baku pemrogramman linier. Bentuk baku atau standar linier programming pada tahap awal ini tidak hnya merubah semua bentuk persamaan kendala menajd persamaan murni berbentuk sama dengan (=), namun juga memastikan bahwa fungsi persamaan kendala terdapat variabel basis awal yang diperlukan saat melakukan iterasi pertama tabel simpleks. Variabel basis awal yang akan dipakai di tabel simpleks awal masih berstatus sumber daya yang masih belum dilakukan aktivitas perubahan sehingga variabel keputusan yang terdapat di tabel simpleks awal masih bernilai nol (0). Hal ini penting dipahami saat memasukkan nilai-nilai yang diperlukan ditabel simpleks awal terutama pada sel variabel keputusan.

Berikut ini summary dari beberapa hal yang harus diperhatikan dalam membentuk persaman program linier yaitu:

1. Fungsi persamaan setiap kendala atau sumber daya dengan bentuk pertidaksamaan (≤) harus diubah ke dalam bentuk murni persamaan (=) dengan menambahkan ruas kiri berupa variabel slack atau *slack variable* yang bertanda positif (+)
2. Fungsi persamaan setiap kendala atau sumber daya dengan bentuk pertidaksamaan (≤) harus diubah ke dalam bentuk murni persamaan (=) dengan dengan mengurangkan satu variabel surplus di ruas kiri kendala. Setelah menjadi bentuk persamaan, perlu di cek lagi apakah kendala tersebut sudah ada variabel basisnya, jika tidak ada ditambahkan variabel basis semu.
3. Fungsi persamaan kendala atau sumber daya yang sudah berbentuk standard persamaan perlu ditambahkan sebuah variabel basis di ruas kiri kendala yang dianggap sebagai variabel *artificial* (buatan).

Dari contoh kasus 1, dapat kita selesaikan dengan metode simpleks. Dari bentuk umum yang sudah diselesaikan sebelumnya, akan kita ubah ke dalam bentuk baku, maka model matematikanya akan berubah seperti berikut.

Fungsi tujuan

$$MAX\ z = 20P1 + 60P2 + 0S1 + 0S2 + 0S3 + 0S4$$

Menjadi

$$z - 20P1 - 60P2 + 0S1 + 0S2 + 0S3 + 0S4 = 0$$

Fungsi kendala

Kendala jam tenaga kerja

$$30P1 + 20P2 + S1 = 2700$$

Kendala jam kerja mesin

$$5P1 + 10P2 + S2 = 850$$

Kendala jumlah produksi minimal

$$P1 + P2 - S3 + S4 = 95$$

Kendala variabel non negatif

$$P1, P2, S1, S2, S3, S4 \geq 0$$

S1 dan S2 merupakan variabel *slack*, sedangkan S3 merupakan variabel surplus dan S4 merupakan artifisial

Di dalam metode simpleks, perhitungan iterative akan menggunakan alat bantu tabel simpleks. Bentuk baku atau standar yang telah dikondisikan di awal harus dimasukkan nilai koefisiennya saja ke dalam tabel simpleks. Semua bentuk variabel keputusan yang bukan variabel basis punya nilai kanan (solusi kanan) sama dengan nol (0) dan koefisien variabel basis pada baris fungsi baris tujuan harus sama dengan 0. Sehingga saat membentuk tabel simpleks awal, variabel basis menjadi acuan pertama yang sebaiknya ditulis lebih dulu di tabel simpleks awal. Dalam bab ini fngsi kendala yang memakai variabel *slack* dalam bentuk standar atau baku akan menjadi fokus bahasan.

Berikut ini tahap-tahap penyelesaian iterative metode simpleks:

1. Iterasi 0. Sebelum dilanjutkan ke proses berikutnya, periksalah kelayakan tabel awal simpleks. Kelayakan tabel awal simpleks bisa dilihat dari nilai ruas kanan. Apabila solusi ada yang bernilai negative (-), maka proses tidak dapat dilanjutkan ke iterasi berikutnya dan solusi optimal tidak bisa dicari.

Tabel 3. 2 Iterasi 0

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-20	-60	0	0	0	0		
S1	30	20	1	0	0	0	2700	
S2	5	10	0	1	0	0	850	
S4	1	1	0	0	-1	1	95	

2. Langkah berikutnya adala menentuka kolom pivot. Keputusan untuk menentukan kolom kunci (pivot) dilihat dari nilai koefisien dari fungsi tujuan (nilai sebelah kanan baris Z) yang sangat tergantung dari bentuk fungsi tujuan. Apabila fungsi tujuan berbentuk maksimasi, maka kolom pivot akan dipilih dari koefisien yang bernilai paling besar negatifnya (negative terbesar di baris itu). Namun apabila fungsi tujuan berbentuk minimasi, maka sebaliknya kolom kunci akan dipilih berdasarkan nilai koefisien poistif. Apabila kolom pivot yang sudah terpilih ditandai sepanjang kolom tersebut, maka variable keputusan yang berkorespondensi dengan kolom pivot tersebut akan kita jadikan sebagai variabel keluar (*outcomin variable*). Untuk beberapa kasus tertentu, kadang kita jumpai nilai paling negative untuk fungsi tujuan maksimasi atau nilai positif terbesar untuk fungsi tujuan minimasi terdapat nilai yang sama di kolom yang berbeda, maka untuk kasus tersebut kita bisa memilh sembarang kolom pivot.

   Dari tabel dapatkan nilai negative paling besar berada di posisi kolom P2, maka kolom P2 adalah kolom pivot, dan P2 adalah variabel masuk.

Tabel 3. 3 Kolom Pivot

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-20	-60	0	0	0	0		
S1	30	20	1	0	0	0	2700	
S2	5	10	0	1	0	0	850	
S4	1	1	0	0	-1	1	95	

3. Pada langkah ini, tentukanlah baris kunci atau pivot. Baris kunci ini ditentukan dengan cara membagi nilai nilai di kolom solusi dengan nilai dengan masing-masing nilai yang ada di seluruh kolom pivot yang saling berkorespondensi yang terletak di dalam baris yang sama. Dalam proses pembagian tersebut nilai nol atau negative diabaikan dan tidak diukutkan menjadi pembagi. Dari hasil pembagian tersebut, maka baris pivot dipilih dari baris dengan rasio pembagian paling kecil. Sehingga, apabila baris pivot sudah ditentukan, maka variabel keluar bisa ditentukan dari variabel yang berada di kolom paling kiri di baris pivot tersebut. Untuk kasus tertentu, kadang dijumpai nilai hasil pembagian yang sama berjumlah lebh dari satu. Untuk kasus tersebut maka tidak ada aturan baku menentukan pilihan jadi pilihlah salah satu secara sembarang.

Tabel 3. 4 Rasio

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-20	-60	0	0	0	0		
S1	30	20	1	0	0	0	2700	135
S2	5	10	0	1	0	0	850	85
S4	1	1	0	0	-1	1	95	95

Dari tabel di atas, maka bisa dilihat hasil pembagian atau rasio pembagian nilai ruas kanan tabel simpleks dengan nilai di kolom kunci adalah 85 yang berkorespondensi/ bersesuaian dengan baris S2. Rasio pembagian nilai kanan dengan kolom pivot terkecil adalah 85 bersesuaian dengan baris S2, sehingga baris S2 merupakan baris kunci (pivot) yang berarti variabel S2 akan menjadi variabel yang keluar (*outcoming variable*) pada iterasi selanjutnya metode simpleks.

4. Tentukan elemen pivot. Elemen pivot merupakan nilai yang terletak pada perpotongan kolom dan baris pivot. Elemen pivot adalah 10.

Tabel 3. 5 Baris Pivot dan Elemen Pivot Iterasi 0

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-20	-60	0	0	0	0		
S1	30	20	1	0	0	0	2700	135
S2	5	10	0	1	0	0	850	85
S4	1	1	0	0	-1	1	95	95

5. Pada tahap berikutnya adalah pembentukan tabel simpleks yang baru. Tabel simpleks yang baru, dibentuk dengan mempertimbangkan nilai baris pivot yang baru. Baris pivot yang baru merupakan baris pivot yang lama (sebelumnya) yang dibagi dengan elemen-elemen pivot. Sementara nilai baris baru yang lainnya dihitung dengan dengan mengurangkan nilai di kolom pivot baris yang bersangkutan yang telah dikalikan dengan nilai baris pivot baru dalam satu kolom terhadap nila-nilai baris lamanya (sebelumnya) yang terletak di kolom tersebut.

   Dari hasil perhitungan, kita dapatkan nilai-nilai pertama yang dimiliki adalah nilai-n pada baris kunci baru yang telah dikali dengan 2. Berikutnya, semua nilai baris S2 pada tabel solusi awal sebelumnya atau iterasi ke-0 dibagi dengan nilai 10 (elemen kunci atau pivot)

Tabel 3. 6 Iterasi 1

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z								
S1								
S2	5/10	10/10	0/10	1/10	0/10	0/10	850/10	
S4								

Perhitungan untuk baris Z

-20	-60	0	0	0	0	0	
-60	(5/10)	(10/10)	(0/10)	(1/10)	(0/10)	(0/10)	(850/10) –
	-10	**0**	**0**	**6**	**0**	**0**	**5100**

Perhitungan untuk baris S1

	30	20	1	0	0	0	2700
20	(5/10)	(10/10)	(0/10)	(1/10)	(0/10)	(0/10)	(850/10) –
	10	**0**	**1**	**-2**	**0**	**0**	**-1000**

Perhitungan untuk baris S4

	1	1	0	0	-1	1	95
1	(5/10)	(10/10)	(0/10)	(1/10)	(0/10)	(0/10)	(850/10) –
	0,5	**0**	**0**	**-0,1**	**-1**	**1**	**10**

Maka tabel iterasi 1 ditunjukkan pada tabel 7

Tabel 3. 7 Hasil Iterasi 1

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-10	0	0	6	0	0	5100	
S1	20	0	1	-2	0	0	1000	
S2	0,5	0	0	-0,1	-1	1	10	
S4	0,5	0	0	-0,1	-1	1	10	

6. Langkah berikutnya adalah memeriksa tabel simpleks untuk mengetahui apakah tabel sudah optimal. Kondisi optimal tabel simpleks dapat dilihat dari koefisien fungsi tujuan yang direpresentasikan oleh nilai di baris Z yang juga angat tergantung dari bentuk persamaan fungsi tujuan. Untuk persamaan fungsi tujuan yang maksimasi tabel simpleks dikatakan sudah optimal dan tidak perlu dilanjutkan lagi ke iterasi selanjutnya jika semua nilai yang ada di baris Z sudah nol atau positif. Sementara pada persamaan fungsi tujuan minimasi, tabel simplkes dikatakan optimal dan dihentikan prosesnya jika semua nilai pada baris Z sudah bernilai negative atau nol. Apabila nilai-nilai yang ada di baris Z belum memenuhi aturan di atas maka langkah berikutnya harus kembali ke langkah nomor 2 sampai didapatkan kondisi optimal.

Seperti langkah sebelumnya, di kasus yang sama di tahap ini kita perlu memeriksa tabel simpleks apakah sudah optimal atau belum.

Oleh karena nilai-nilai di baris Z masih bernilai negative maka tabel simpleks belum optimal dan harus dilanjutkan ke iterasi berikutnya. Kolom kunci dan baris kunci diberi tanda seperti sebelumnya.

Tabel 3. 8 Kolom, Baris dan Elemen Pivot Iterasi 1

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	-10	0	0	6	0	0	5100	-510
S1	20	0	1	-2	0	0	1000	50
S2	0,5	0	0	-0,1	-1	1	10	20
S4	0,5	0	0	-0,1	-1	1	10	20

Variabel masuknya adalah P1, variabel keluarnya adalah S4, dan elemen pivotnya adalah 0,5. Hasil perhitungan pada iterasi 2 bisa dilihat di tabel berikut:

Tabel 3. 9 Iterasi 2

VB	P1	P2	S1	S2	S3	S4	NK	Rasio
Z	0	0	0	2	-20	20	4900	
S1	0	0	1	2	-40	-40	600	
S2	0	1	0	0,2	1	-1	75	
S4	1	0	0	-0,2	-2	2	20	

Dari hasil perhitungan, didapatkan tabel simpleks di atas yang sudah optimal. Hal trsebut dapat dilihat dari semua nilai yang ada di baris Z yang sudah bernilai positif atau nol sehingga proses perhitungan dihentikan di tahap ini dan tidak diperlukan iterasi selanjutnya. Perlu diingat bahwa perhitungan dalam metode simpleks memerlukan tingkat keakuratan dan ketelitian yang sangat tinggi karena saling terkait antara satu iterasi dengan iterasi berikutnya. Apalagi saat menghadapi nilai-nilai yang berupa nilai decimal atau pecahan, sehingga proses pembulatan nilai harus dilakukan dengan hati-hati. Sangat disarankan agar bilangan dibiarkan pecahan bukan dalam bentuk decimal agar lebih mudah dikerjakan. Ketidaktelitian dalam proses perhitingan dalam metode simpleks akan menyebabkan proses iterasi akan bertambah panjang bahkan berisiko tidak selesai atau tidak didapatkan nilai optimalnya.

Perhitungan iterative dalam metode simpleks pada intinya adalah melakukan pengecekan nilia satu per satu titik-titik ekstrim yang ditemukan yang berada di daerah penyelesaian yang layak. Proses pemeriksaan dimulai dari kondisi nol saat tabel awal simpleks dibuat yang menerangkan bahwa semua aktivitas belum dimulai sehingga variabel keputusan masih bernilai nol. Sehingga apabila jumlah titik-titik ektrim berjumlah lebih dari satu atau n, maka kemungkinan besar jumlah iterasinya dalam proses perhitungan juga berjumlah n kali.

Pada tabel iterasi 2 dapat dilihat bahwa solusi optimal adalah pada NK atau sisi kanan, untuk P1 adalah 20 ton, P2 = 75 ton dan keuntungan maksimal atau Z adalah 4.900.

BAB 4
MASALAH PENUGASAN

Aplikasi Metode Penugasan
Langkah Penyelesaian Metode Penugasan: Minimasi
Penyelesaian Masalah Penugasan dengan R

MASALAH PENUGASAN

Pada bab sebelumnya telah dibahas mengenai metode transportasi sebagai tipe khusus dari pemrograman linier. Pada sub bab ini akan dibahas suatu tipe khusus dari masalah transportasi, yaitu masalah penugasan (*assignment problem*).

Dalam permasalahan penugasan ini akan ditentukan solusi sejumlah tugas yang akan ditugaskan kepada penerima tugas tersebut. Jika terdapat 10 penerima tugas maka akan dibuat 10 tugas yang akan dibagi satu per satu kepada penerima tersebut. Sehingga, pada permasalahan penugasan ini jumlah tugas akan menyesuaikan jumlah dari penerima tugas tersebut. Dengan begitu sudah pasti kita harus mengetahui data apa saja tugas yang harus dikerjakan dan ada berapa banyak sumber daya yang dapat diberikan tugas. Data tersebut merupakan data pokok sebelum akhirnya tugas dapat dibagi kepada masing-masing penerima tugas. Selain itu, kita juga memerlukan data yang berhubungan dengan jumlah kerugian yang ditimbulkan atau jumlah keuntungan yang diperoleh jika penerima tugas menyelesaikan setiap tugas yang diterimanya.

Tujuan utama yang akan dicapai pada masalah penugasan ini adalah mendapatkan keuntungan sebesar-besarnya dan kerugian sekecil-kecilnya. Hal tersebut diharapkan dapat tercapai dengan membagi sejumlah tugas kepada sejumlah penerima tugas secara efektif dan efisien. Dapat dilihat bahwa secara umum maalah penugasan terbagi menjadi dua yaitu masalah yang berhubungan dengan maksimasi dan minimasi. Penyelesaian permasalahan penugasan umumnya dapat dilakukan dengan beberapa metode.

Menurut referensi terdahulu metode transportasi cukup baik untuk menyelesaikan permasalahan ini, tetapi metode hongaria dinilai lebih baik untuk permasalahan ini.

Metode Hongaria memiliki prinsif kerja memanipulasi data matriks biaya yang telah dikumpulkan atau diketahui sebelumnya. Manipulasi dilakukan dengan cara mencari hasil pengurangan masing-masing elemen baris dengan elemen yang paling minimum pada baris itu sendiri. Setelah itu dilanjutkan dengan mencari hasil pengurangan masing-masing elemen kolom dengan elemen yang paling minimum pada kolom tersebut. Langkah selanjutnya adalah menandai elemen-elemen yang bernilai 0 dengan menghubungkannya dengan elemen bernilai 0 lainnya sehingga terbentuk garis penghubung. Setelah langkah tersebut selesai dilakukan, maka perhatikan sub matriks yang tidak dilewati garis kemudian cari elemen minimumn 23 Langkah terakhir adalah dengan mengurangkan elemen minimum dari setiap elemen pada submatriks yang tidak dilewati garis dan ditambahkan pada elemen yang dilalui dua garis. Langkah-langkah manipulasi tersebut dilakukan berulang kali hingga mat 81 s biaya yang optimum diperoleh. Matriks biaya optimum selesai saat banyaknya garis (yang melalui elemen '0') berjumlah sama dengan n.

Apabila permasalahan yang diselesaikan memiliki tujuan minimasi biaya maka penyelesaian dapat dilakukan secara langsung. Sedangkan apabila permasalahan yang diselesaikan memiliki tujuan maksimasi profit maka penyelesaian dapat dilakukan dengan meminimumkan negatif dari biaya.

Aplikasi Metode Penugasan

Ada begitu banyak aplikasi metode penugasan yang dapat dilakukan untuk menghadapi permasalahan yang terkait dengan penentuan keputusan misalnya sebagai berikut:

1. Menetukan daerah perdagangan/penjualan
2. Layout fasilitas untuk penanganan material
3. Penjadwalan produksi
4. Penugasan karyawan
5. Pemilihan supplier
6. Penentuan kontrak

Pada umumnya masalah penugasan (*assignment problem*) punya tujuan untuk mencari keputusan optimal "siapa mengerjakan apa" jika terkait dengan sumber daya yang dimiliki dengan tujuan utama memaksimalkan pendapatan (bisa profit, perolehan penjualan maupun waktu penyelesaian) atau meminimalkan pengeluaran (bisa berupa biaya maupun waktu penyelesaian).

Contoh Masalah Penugasan: Minimasi

Sebuah perusahaan memiliki 3 staf ahli yang saat ini bekerja di 3 cabang perusahaan. Mereka akan dimutasi ke tiga cabang perusahaan lain yan membutuhkan kinerja mereka. Bantulah bagian personalia untuk menugaskan staf ahli tersebut sedemikan rupa sehingga satu staf ahli hanya untuk satu cabang perusahaan dengan meminimalkan biaya perjalanan (biaya dalam satuan jutaan).

		Tujuan Mutasi		
		1	2	3
Cabang Sebelumnya	A	20	30	40
	B	10	20	25
	C	20	15	10

Gambar 4. 1 Matriks Biaya Perjalanan

Contoh Masalah Penugasan: Maksimasi

Selain dari bagaimana meminimumkan biaya perjalanan, bagian personalia perusahaan tersebut juga memiliki perkiraan kinerja setiap staf ahli untuk setiap cabang yang akan ditempati. Bantulah bagian personalia untuk menentukan tempat mutasi terbaik bagi setiap staf ahli agar keuntungan yang didapat oleh perusahaan menjadi maksimum.

		Tujuan Mutasi		
		1	2	3
Cabang Sebelumnya	A	20	30	40
	B	10	20	25
	C	20	15	10

Gambar 4. 2 Perkiraan Kinerja

Langkah Penyelesaian Metode Penugasan: Minimasi

Adapun contoh penyelesaian masalah penugasan dengan fungsi tujuan minimasi (metode Hungarian) adalah sebagai berikut (lihat kembali contoh minimasi pada sub bab sebelumnya).

Menyusun tabel biaya seperti yang sudah ditunjukkan pada gambar sebelumnya.

Langkah pertama adalah menentukan nilai minimal setiap baris yang ada sebelum akhirnya dilakukan proses pengurangan nilai setiap baris dengan nilai minimum baris.

Untuk kasus minimasi di atas, kita mempunyai biaya terkecil untuk setiap baris adalah

A = 20,

B = 10,

C = 10

Pengurangan baris dari cabang A akan diperoleh angka sebagai berikut.

20 – 20 = 0

30 – 20 = 10

40 – 20 = 20

Hasil pengurangan baris ini dapat dilihat pada tabel berikut. Sedangkan pengurangan baris untuk B dan C dapat anda hitung sendiri.

		Tujuan Mutasi		
		1	2	3
Cabang Sebelumnya	A	0	10	20
	B	0	10	15
	C	10	5	0

Gambar 4. 3 Hasil Pengurangan Baris

Nilai yang terdapat pada baris baru atau pada tabel di atas adalah *opportunity cost*.

1. Pada tahap ini lakukan pemeriksaan apakah setiap angka 0 telah ada pada setiap kolom. Jika angka 0 telah ada pada setiap kolom maka kita dapat melanjutkan ke langkah 4. Namun, jika angka 0 belum ada pada setiap kolom maka harus ditentukan nilai terkecil dari setiap kolom. Pada tahap ini cukup kolom yang belum memiliki angka 0 saja yang dicari nilai terkecilnya. Kemudian lakukan pengurangan setiap angka pada kolom tersebut dengan nilai terkecil yang telah dicari sebelumnya.

   Pada kasus ini, dapat dilihat pada kolom 2 tidak ada angka 0. Sehingga carilah angka terkecil pada kolom tersebut, yaitu adalah angka 5. Sehingga pengurangan kolom 2 sebagai berikut.

   $10 - 5 = 5$

   $10 - 5 = 5$

   $5 - 5 = 0$

		Tujuan Mutasi		
		1	2	3
Cabang Sebelumnya	A	0	5	20
	B	0	5	15
	C	10	0	0

Gambar 4. 4 Hasil Pengurangan Kolom

2. Tentukan apakah terdapat baris yang memiliki sejumlah angka nol, namun tidak 2 angka nol pada satu baris atau kolom sama. Tabel dikatakan optimal saat ada 2 angka nol pada satu baris atau kolom yang sama. Jika tidak, maka harus melanjutkan ke langkah 5.

   Langkah ini adalah memeriksa apakah suatu penugasan sudah layak atau belum. Jika diperhatikan akan ditemui sejumlah 4 angka nol pada tabel 4. Jika misalnya kita ambil 3 angka nol, masih tidak ada yang berada pada baris/kolom yang berbeda. Misalnya pada kotak (1,1) (3,2) dan (3,3); kita masih menemukan angka nol pada baris yang sama, yaitu pada (3,2) dan (3,2), begitu juga dengan kemungkinan lainnya. Hal tersebut menandakan tabel belum dapat dikatakan optimal, sehingga untuk menghasilkan tabel optimal langkah selanjutnya harus dilakukan.

3. Berikan garis horizontal atau vertical seminimal mungkin pada kolom atau tabel yang bernilai nol. Penugasan optimal dapat dibentuk apabila jumlah garis minimum sama dengan jumlah baris/kolom.

Cabang Sebelumnya	Tujuan Mutasi 1	2	3
A	0	5	20
B	0	5	15
C	10	0	0

Gambar 4. 5 Penentuan Jumlah Garis Minimal

Jumlah dari garis minimal adalah 2 namun jumlah baris/kolom adalah 3 sehingga penugasan optimal belum dapat ditentukan. Sehingga dilanjutkan ke langkah 6.

4. Carilah nilai paling minimum di antara seluruh nilai yang belum tertutup garis. Kemudian kurangkan masing-masing nilai yang tidak tertutup garis dengan nilai paling minimum yang telah dicari sebelumnya. Untuk nilai yang tertutupi 2 garis tambahkan dengan nilai paling minimum yang telah dicari sebelumnya.

   Angka 5 merupakan angka yang paling minimal dari seluruh angka yang belum dilewati garis. Sehingga tabel akan menjadi seperti Gambar 4.6.

Cabang Sebelumnya	Tujuan Mutasi 1	2	3
A	0	0	15
B	0	0	10
C	15	0	0

Gambar 4. 6 Pengurangan Nilai yang Tidak Dilalui Garis

5. Pada langkah ini periksalah apakah penugasan yang optimal telah terbentuk. Untuk lebih detailnya silahkan lihat pada Gambar 4.7.

Cabang Sebelumnya	Tujuan Mutasi 1	2	3
A	0	0	15
B	0	0	10
C	15	0	0

Gambar 4. 7 Penentuan Garis Minimal

Garis yang menghubungkan setiap angka nol berjumlah 4 garis (dengan minimal jumlah garis = 3 karena jumlah baris/kolom = 3), dan dapat kita lihat juga angka nol berada pada baris dan kolom yang berbeda. Sehingga tabel sudah optimal.

6. Selanjutnya, akan dilakukan penugasan optimum. Dimana kita akan melihat pada angka nol pada setiap baris dan kolom. Mungkin terdapat lebih dari satu cara penugasan optimal, dalam kasus seperti ini disebut multi opimum.

   Pada cabang A, staf ahli dapat dipindahkan ke tujuan 1 dan 2

   Pada cabang B, staf ahli dapat dimutasi ke tujuan 1 dan 2

   Dan staf ahli pada cabang C dapat dimutasi ke tujuan 2 dan 3

   Skenario 1.

   Staf ahli A dimutasi ke tujuan 1

   Staf ahli B dimutasi ke tujuan 2

   Staf ahli C dimutasi ke tujuan 3

   Perhitungan biaya perjalanan = 20 + 20 + 10 = 50

   Skenario 2.

   Staf ahli A dimutasi ke tujuan 2

   Staf ahli B dimutasi ke tujuan 1

   Staf ahli C dimutasi ke tujuan 3

   Perhitungan biaya perjalanan = 30 + 10 + 10 = 50

Dapat dilihat pada masing-masing skenario bahwa biaya perjalanan akan sama dengan 50, sehingga dapat dilakukan untuk skenario 1 ataupun 2.

Masalah-masalah Khusus dalam Metode Penugasan

1. **Masalah maksimasi keuntungan**

   Jika akan mencari hasil dari maksimasi keuntungan, maka kalikan keuntungan dengan minus satu (-1). Sebagai contoh pada kasus 2, jika penugasan keuntungan 20, maka biayanya berubah menjadi -20. Sehingga, biaya minimum setiap baris atau kolom adalah yang memiliki jumlah negatif terbesar.

2. **Masalah tidak seimbang**

   Jika terjadi jumlah baris (pekerja) lebih besar dari jumlah kolom (pekerjaan) ataupun sebaliknya, dapat dikatakan bahwa penugasan ini adalah penugasan yang tidak seimbang. Sehingga untuk menyelesaikannya adalah dengan cara menambahkan pekerja ataupun pekerjaan semu, sehingga matriks penugasan menjadi seimbang. Setiap pekerja atau pekerjaan semu tersebut diberi nilai 0.

3. **Masalah pemblokiran dan prioritas**

   Masalah pemblokiran adalah untuk mencegah agar suatu pekerja tidak ditugaskan untuk menyelesaikan pekerjaan tertentu. Karena penugasan biasanya untuk mencari biaya minimum, masalah pemblokiran dilakukan dengan memberi koefisien biaya yang sangat besar atau M. Sedangkan untuk masalah prioritas merupakan kebalikan dari masalah pemblokiran. Prioritas berarti mengutamakan pekerja tertentu untuk menyelesaikan pekerjaan tertentu. Untuk melakukan hal ini, maka koefisien biaya prioritas diberi nilai 0.

Penyelesaian Masalah Penugasan dengan R

Kasus Minimasi

Pada R, kita dapat menyelesaiakan masalah penugasan dengan menggunakan package lpSolve (yang sudah dijelaskan pada bab 2).

Sama halnya dengan lp dan lp.transport, fungsi lp.assign adalah script program. Dapat kita lihat dengan mengetikkan lp.assign di console.

```
"> lp.assign
function (cost.mat, direction = "min", presolve = 0, compute.sens = 0)
{
    if (!is.matrix(cost.mat))
        stop("Matrix of costs required.")
```

```
if (is.data.frame(cost.mat))
   cost.mat <- as.matrix(cost.mat)
nr <- nrow(cost.mat)
nc <- ncol(cost.mat)
rnum.signs <- rep(3, nr)
row.rhs <- rep(1, nr)
cnum.signs <- rep(3, nc)
col.rhs <- rep(1, nc)
if (direction == "min")
   direction <- as.integer(0)
else if (direction == "max")
   direction <- as.integer(1)
else stop("Direction must be 'min' or 'max'")
varcount <- as.integer(nr * nc)
objective <- as.double(c(0, c(t(cost.mat))))
const.count <- as.integer(nr + nc)
intcount <- as.integer(varcount)
intvec <- as.integer(1:varcount)
objval <- as.double(0)
int.count <- nc * nr
integers <- as.integer(numeric(int.count))
solution <- as.double(numeric(nc * nr))
status <- as.integer(0)
sens.coef.from <- sens.coef.to <- 0
duals <- duals.from <- duals.to <- 0
if (compute.sens) {
   sens.coef.from <- sens.coef.to <- numeric(varcount)
   duals <- duals.from <- duals.to <- numeric(varcount
+
      const.count)
}
```

```
        lps.out <- .C("lp_transbig", direction = direction, rcount =
as.integer(nr),
        ccount = as.integer(nc), costs = objective, rsigns = as.integer(rnum.
signs),
        rrhs = as.double(row.rhs), csigns = as.integer(cnum.signs),
        crhs = as.double(col.rhs), objval = objval, int.count = int.count,
      integers = integers, solution = solution, presolve = as.integer(presolve),
        compute.sens = as.integer(compute.sens), sens.coef.from =
as.double(sens.coef.from),
        sens.coef.to = as.double(sens.coef.to), duals = as.double(duals),
       duals.from = as.double(duals.from), duals.to = as.double(duals.to),
        status = status, PACKAGE = "lpSolve")
     lps.out$solution = matrix(lps.out$solution, nr, nc, byrow = TRUE)
     if (length(duals) > 0) {
        lps.out$sens.coef.from <- matrix(lps.out$sens.coef.from,
          nr, nc, byrow = TRUE)
        lps.out$sens.coef.to <- matrix(lps.out$sens.coef.to,
          nr, nc, byrow = TRUE)
        lps.out$duals <- matrix(lps.out$duals, nr, nc, byrow = TRUE)
        lps.out$duals.from <- matrix(lps.out$duals.from, nr,
          nc, byrow = TRUE)
        lps.out$duals.to <- matrix(lps.out$duals.to, nr, nc,
          byrow = TRUE)
     }
     lps.out$costs <- cost.mat
     if (any(names(version) == "language"))
        class(lps.out) <- "lp"
     else oldClass(lps.out) <- "lp"
     lps.out
  }"
```

Untuk penugasan, kita akan menggunakan fungsi lp.assign dimana fungsi ini penggunaannya seperti berikut.

lp.assign (cost.mat, direction = "min")

cost.mat adalah matriks biaya

direction adalah fungsi tujuan, dapat berupa "min" untuk minimasi (default) dan "max" untuk maksimasi.

Contoh penyelesaian kasus 1.

```
# Memanggil package lpSolve
library(lpSolve)
# Membuat matriks biaya penugasan
assign.costs <- matrix (c(20,10,20,30,20,15,40,25,10),  3,3)
# Menampilkan solusi dari masalah penugasan
lp.assign (assign.costs)$solution
# Menampilkan hasil dari fungsi tujuan
lp.assign (assign.costs)
```

Pada bagian # Membuat matriks biaya penugasan, data yang digunakan adalah data kasus 1. Angka 3,3 pada bagian ujung dari variabel assign.cost menandakan bahwa matriks yang dibuat berordo 3x3. Ketika kita memanggil variabel assign.cost, maka akan menampilkan matriks sebagai berikut.

```
     [,1] [,2] [,3]
[1,]   20   30   40
[2,]   10   20   25
[3,]   20   15   10
```

Pada bagian # Menampilkan solusi dari masalah penugasan akan menampilkan tabel optimal. Ketika fungsi lp.assign (assign. costs)$solution dijalankan, maka akan menampilkan solusi dari masalah penugasan kasus 1

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Angka 1 di atas menandakan bahwa baris 1 (dalam kasus adalah staf ahli A) akan ditempatkan di kolom 1 (dalam kasus adalah tujuan ke 1).

Begitu seterusnya seperti skenario 1 pada kasus 1 yang sudah dijelaskan di sub bab sebelumnya.

Fungsi lp.assign (assign.costs) ketika dijalankan akan menghasilkan fungsi tujuan. Dalam kasus 1, berarti adalah biaya minimal dari masalah penugasan ini. Ketika fungsi tersebut dijalankan, akan menampilkan hasil sebagai berikut.

Success: the objective function is 50

Yang berarti adalah biaya minimal adalah senilai 50.

Penyelesaian Masalah Penugasan dengan R: Maksimasi

Setelah kita mengetahui bagaimana menyelesaikan masalah penugasan untuk minimasi biaya, selanjutnya kita akan menyelesaikan masalah penugasan untuk kasus maksimasi keuntungan. Dengan menggunakan data kasus 2 pada sub bab sebelumnya, maka penyelesaiannya adalah sebagai berikut.

```
assign.costs <- matrix (c(20,10,20,30,20,15,40,25,10), 3,3)
lp.assign (assign.costs, direction = "max")$solution
lp.assign (assign.costs, direction = "max")
```

dengan menggunakan langkah penyelesaian seperti script di atas, maka hasil dari kasus 2 adalah sebagai berikut.

```
[,1] [,2] [,3]
[1,]    0    0    1
[2,]    0    1    0
[3,]    1    0    0
```

Staf ahli A akan dimutasi ke tujuan ke 3, staf ahli B akan dimutasi ke tujuan ke 2 sedangkan staf ahli C akan dimutasi ke tujuan ke 1. Dengan penugasan tersebut, akan menghasilkan keuntungan maksimal sebanyak 80 (dengan menggunakan lp.assign (assign.costs, direction = "max")).

BAB 5
PEMROGRAMAN LINIER INTEGER

Aplikasi Integer Linear Programing
Metode Pemecahan Pemrograman Integer
Penyelesaian Masalah ILP Dengan R

PEMROGRAMAN LINIER INTEGER

Pemrograman linier yang merupakan dasar dari Pemrograman linier integer (*integer linear programming*/ILP) biasanya digunakan untuk mengalokasikan sumber daya untuk mencapai hasil yang optimal misalnya dalam hal memaksimumkan keuntungan atau meminimumkan biaya operasional. Oleh karena itu program linier biasanya digunakan untuk mengatasi permasalahan dalam industri. Namun hasil yang diperoleh tidak selalu berbentuk integer, padahal hasil yang dibutuhkan harus berbentuk integer misalnya jumlah karyawan atau mesin yang optimal untuk sebuah perusahaan. Solusi yang sering dilakukan yaitu dengan melakukan pembulatan hasil tersebut ke nilai integer yang terdekat dan layak. Hal seperti itu bisa saja dilakukan, tapi masalah lain yang muncul yaitu tidak ada jaminan bahwa hasil yang sudah dibulatkan memenuhi batasan yang ada dan layak untuk digunakan. Misalnya jika variabel keputusan berkaitan dengan jumlah mesin yang digunakan untuk produksi dan diperoleh hasil yang optimum yaitu 13/2, maka tidak mungkin memproduksi mesin dengan jumla yang dihasilkan sehingga keputusan yang bisa di ambil yaitu memproduksi 6 atau 7 mesin.

Ketidaklayakan dari hasil pembulatan bisa ditolerir untuk beberapa kasus yang parameter (yang diestimasi) dari masalah tersebut tidak pasti. Terdapat batasan untuk permasalahan integer yang menggunakan parameter pasti misalnya $x_1+x_2+x_3+\ldots+x_n=1$, dimana $x_j = (0,1)$ untuk semua j. Untuk kasus seperti itu, pembulatan tidak dapat digunakan dan algoritma pasti yang menjadi sangat penting. Lebih lanjut, ketidaklayakan pembulatan bukan hanya mewakili objek yang bersifat diskrit (misalnya

orang, mesin, dll) tapi juga keputusan lain yang menggunakan variable biner (x=0 jika ditolak dan x=1 jika diterima) misalnya pada keputusan pendanaan suatu proyek. Sehingga hal ini menunjukan bahwa dibutuhkan program linier yang dapat digunakan untuk mengatasi permasalahan seperti itu.

Pemrograman linier integer (*integer linear programming*/ILP) pada dasarnya berkaitan dengan program linier yang dimana beberapa atau semua variabelnya memiliki nilai yang integer (bulat) atau diskrit. ILP bisa bersifat murni atau campuran tergantung pada beberapa atau semua variabelnya dibatasi oleh nilai integer. Untuk memperjelas pentingnya dan bagaimana penggunaan ILP, berikut disajikan penggunaan ILP dalam berbagai permasalahan.

Aplikasi Integer Linear Programing

Setelah memahami definisi dan pentingnya ILP, selanjutnya pada bagian berikut disajikan beberapa aplikasi ILP pada beberapa permasalahan.

1. Masalah Penganggaran Modal

Keputusan mengenai modal dalam sebuah perusahaan merupakan hal yang medasar karena hal ini berkaitan dengan proses perencanaan, penetapan tujuan dan prioritas, pengalokasian dana dan penentuan kriteria pemilihan yang sifatnya jangka panjang misalnya mesin-mesin yang akan digunakan dalam produksi. Kesalahan dalam pengambilan keputusan akan berdampak besar terhadap kelangsungan hidup perusahaan (Widodo, 2013). Misalnya kesalahan pemilihan mesin produksi yang bekerja lebih lamban dibandingkan mesin produksi yang dimiliki perusahaan pesaing sehingga berdampak pada kecepatan produksi yang meyebabkan perusahaan kalah saing dengan perusahaan pesaing.

Contoh yang akan dibahas pada bagian ini yaitu pemilihan proyek yang akan dijalankan dalam 3 tahun mendatang. Contoh ini diambil dari (Taha, 1996). Terdapat lima proyek yang sedang dipertimbangkan untuk dilaksanakan dengan jumlah pengembalian yang diharapkan dari masing-masing proyek dapat dilihat pada tabulasi berikut.

Tabel 5. 1 Jumlah Pengembalian yang Diharapkan

Proyek	Pengeluaran untuk:			Pengembalian
	Tahun 1	Tahun 2	Tahun 3	
1	5	1	8	**20**
2	4	7	10	**40**
3	3	9	2	**20**
4	7	4	1	**15**
5	8	6	10	**30**
Dana yang tersedia	**25**	**25**	**25**	

Permasalahan yang dihadapi yaitu memutuskan yang mana di antara lima proyek tersebut yang harus dilaksanakan dalam periode perencanaan 3 tahun tersebut. Dalam hal ini, keputusan "ya dan tidak" secara numerik akan diberi kode sebagai sebuah variabel biner, dimana 1 mewakili "ya" dan 0 mewakili "tidak". Kemudian formulasikan masalah keputusan ini dengan mendefinisikan variabel biner xj untuk mewakili proyek ke-j. Sehingga diperoleh model:

Maksimumkan $z = 20_{x1} + 40_{x2} + 20_{x3} + 15_{x4} + 30_{x5}$

Dengan batasan :

$$5_{x1} + 4_{x2} + 3_{x3} + 7_{x4} + 8_{x5} \leq 25$$
$$1_{x1} + 7_{x2} + 9_{x3} + 4_{x4} + 6_{x5} \leq 25$$
$$8_{x1} + 10_{x2} + 2_{x3} + {}_{x4} + 10_{x5} \leq 25$$
$$x_j = 0,1, \qquad j = 1,2,\ldots,5$$

Pemecahan kontinyu menggunakan LP yang optimu dengan menggunakan batas atas $x_J \leq 1$ untuk semua j, menghasilkan $x_1 = 0{,}5789$, $x_2 = x_3 = x_4 = 1$, dan $x_5 = 0{,}7386$. Hasil ini sudah pasti tidak bisa digunakan karena bernilai pecahan. Membulatkan hasil ini juga tidak berarti karena x_j diawal sudah didefinisikan sebagai kode numerik untuk keputusan "ya-tidak". Dengan demikian, nilai-nilai pecahan tidak memiliki kaitan dengan model yang telah dirumuskan sebelumny Hasil optimum dengan menggunakan ILP untuk model ini adalah $x_1 = x_2 = x_3 = x_4 = 1$, dengan $x_5 = 0$ dan $z = 105$.

2. Masalah Biaya Tetap

Masalah yang ada dalam perencanaan produksi umumnya melibatkan N produk dan biaya produksi untuk produk j yang terdiri dari biaya tetap K_j yang tidak berantung pada jumlah yang diproduksi serta biaya variabel cj per unit. Sehingga, jika x_j adalah tingkat produksi untuk j, fungsi biaya produksinya yaitu

$$C_j(x_j) = \begin{cases} K_j + c_j x_j, & x_j > 0 \\ 0, & x_j = 0 \end{cases}$$

Kriteria tujuan menjadi

$$\text{Minimumkan } z = \sum_{j=1}^{N} C_j(x_j)$$

Kriteria ini sifatnya adalah nonlinier dalam x_j karena terputus di titik asal. Hal ini menyebabkan z tidak dapat ditelusuri dari sudut pandang analitis. Permasalahan ini dapat ditangani secara analitis dengan memasukkan variabel biner.

$$y_j = \begin{cases} 0, & x_j = 0 \\ 1, & x_j > 0 \end{cases}$$

Kondisi ini dapat dituliskan dalam bentuk satu batasan (linier) sebagai

$$x_j \le M y_j$$

dimana $M > 0$ cukup besar untuk membuat tidak diperlukan dalam kaitannya dengan setiap batasan aktif dalam masalah ini. Sehingga kriteria tujuan tersebut dapat ditulis sebagai

$$\text{Minimumkan } z = \sum_{j=1}^{N} (c_j x_j = K_j y_j)$$

dengan batasan

$0 \le x_j \le M y_j$, semua j

$y_j = 0$ atau 1, semua j.

Untuk menunjukan bahwa $x_j \leq M y_j$ merupakan batasan yang sesuai, perhatikan bahwa jika $x_j > 0$, $y_j = 1$ dan biaya tetap K_j ditambahkan dalam fungsi tujuan. Jika $x_j = 0$, y_j adalah nol atau satu, tetapi karena $K_j > 0$ dan z diminimumkan, y_j sudah pasti berada di tingkat nol.

Menariknya, permasalahan biaya tetap pada awalnya samasekali tidak berkaitan dengan ILP. Namun masalah ini di transformasikan menjadi permasalahan integer nol-satu campuran. Transformasi ini digunakan hanya untuk memudahkan analisis. Kenyataannya, variabel biner yang ditambahkan adalah "ekstra" yang berarti bahwa variabel tersebut tidak mengungkapakn informasi baru tentang solusi pemecahan masalah. Contohnya $y_j = 1$ dalam pemecahan optimal sudah pasti menginformasikan bahwa $x_j > 0$.

3. Masalah Penjadwalan Pekerjaan Pabrik

Masalah yang seringkali terjadi dalam produksi yaitu pengurutan pekerjaan yang melibatkan penyelesaian beberapa operasi (n) yang berbeda di satu mesin dengan waktu sesingkat mungkin. Untuk menjadi produk akhir, bahan baku harus melalui serangkaian operasi yang berbeda dan berurutan. Produk akhir tersebut juga harus selesai pada waktu yang telah ditentukan, tidak boleh terlalu cepat atau terlalu lambat. Dengan demikian, permasalahan tersebut memiliki tiga jenis batasan : (1) pengurutan, (2) tidak adanya interferensi, (3) tanggal selesainya produk. Selain permasalah itu, dua operasi yang diproses secara bersamaan dalam satu mesin yang sama juga tidak boleh terjadi.

Permasalahan dapat dihindari dengan melakukan penjadwalan. Penjadwalan produksi adalah salah satu hal yang penting dalam perusahaan, khususnya yang berbasis pada industry manufaktur yang didalamnya terdapat pengolahan bahan baku menjadi barang jadi dengan menggunakan berbagai macam sumber dan fasilitas yang dimiliki perusahaan (Wildan, Setyanto, & Rahman, 2014). Penjadwalan produksi dilakukan dengan menggunakan metode yang telah dibuat untuk menyelesaikan permasalahan yang berhubungan dengan produk yang akan diproduksi, kapan produk tersebut diproduksi, berapa jumlah produk yang akan diproduksi dan pertanyaan lain yang terkait.

Masalah yang pertama terkait batasan yang ada, dapat dimodelkan seperti berikut. Asumsikan xj adalah waktu untuk operasi awal j, aj adalah waktu pengolahan yang diperlukan untuk menyelesaikan operasi j. Jika operasi i harus mendahului operasi j, batasan pengurutan yang

dihasilkan adalah

$$x_i + a_i \leq x_j$$

Batasan selanjutnya yaitu tidak adanya interferensi, sehingga agar operasi i dan j tidak menggunakan mesin secara bersamaan, maka yang dihasilkan yaitu

$$x_i - x_j \geq a_j \quad \text{atau} \quad x_j - x_i \geq a_i$$

Batasan di atas bergantung secara berturut-turut apakah j mendahului i atau sebaliknya i mendahului i dalam pemecahan optimal.

Adanya batasan "**ini-atau-itu**" (*either or constraints*) menghadirkan permasalahan karena model ini tidak lagi dalam format pemrograman linier. Kesulitan ini dapat diatasi dengan memasukkan variabel biner y_{ij} yang didefinisikan sebagai :

$$y_{ij} = \begin{cases} 0 & \text{jika operasi j mendahului operasi i} \\ 1, & \text{jika operasi i mendahului operasi j} \end{cases}$$

Selanjutnya, karena M cukup besar, batasan "**ini-atau-itu**" menjadi setara dengan kedua batasan ini secara bersamaan

$$My_{ij} + (x_i - x_j) \geq a_j \quad \text{atau} \quad M(1 - y_{ij}) + (x_j - x_i) \geq a_i$$

Transformasi yang dilakukan memberikan arti penting bahwa jika dalam pecahan optimal $y_{ij} = 0$, batasan kedua menjadi belebihan. Sedangkan batasan pertama tetap aktif. Sehingga, jika $y_{ij} = 1$, batasan kedua yang aktif, bukan batasa pertama. Variabel biner yang dimasukkan membuat batasan-batasan tersebut menjadi bentuk integer dimana ILP dapat diterapkan.

Batasan terakhir yaitu tanggal selesinya produk dimana produk tersebut sudah harus siap dikirim. Sehingga diasumsikan bahwa operasi j harus segera diselesaikan sebelum waktu dj, maka

$$x_j + a_j \leq d_j$$

Jika t adalah waktu total yang diperlukan untuk menyelesaikan semua n operasi, masalah yang ada menjadi

$$\text{minimumkan } z = t$$

dengan batasan

$$x_j + a_j \leq t, \quad j = 1, 2, 3, \ldots\ldots, n$$

Metode Pemecahan Pemrograman Integer

Dalam LP, hasil optimum dalam metode simplek diperoleh berdasarkan titik ekstrim dari ruang pemecahan masalah. Penting untuk dipahami bahwa hasil yang diperoleh pada intinya dapat mengurangi usaha pencarian pemecahan optimum yang dilakukan dari sejumlah titik yang tak terbatas menjadi terbatas. Sebaliknya, ILP dimulai dengan titik pemecahan yang terbatas (dengan asumsi ILP murni yang dibatasi). Namun sifat variabel yang bentuknya bilangan bulat menyebabkan perancangan algoritma yang efektif menjadi sulit. Perancangan tersebut bertujuan untuk mencari solusi terbaik di antara titik integer yang layak secara langsung. Kesulitan ini membuat para peneliti mengembangkan prosedur pemecahan berdasarkan LP. Strategi prosedur tersebut diringkas kedalam tiga langkah berikut (Taha, 1996) :

1. Longgarkan ruang pemecahan dari masalah integer yang bersangkutan dengan mengabaikan batasan integer. Langkah ini mengkonversi ILP menjadi LP biasa.
2. Pecahkan model LP "yang longgar" yang dihasilkan dan identifikasi titik optimum (kontinyu) dari LP itu.
3. Dengan dimulai dari titik optimum kontinyu, tambahkan batasan khusus yang akan secara berulang-ulang memaksa titik ekstrim optimum dari model LP yang dihasilkan untuk bergerak ke arah batasan integer yang diinginkan.

Alasan memecahkan permasalahan dengan mencari solusi optimum ILP dengan menggunakan LP karena ada kemungkinan bahwa kedua solusi berdekatan sehingga hal ini meningkatkan kecepatan menemukan solusi masalah integer tersebut. Inti dari prosedur sebelumnya yaitu pendekatan ini memecahkan masalah ILP lebih mudah dibandingkan memcahkan permasalahn ILP secara langsung.

Ada dua metode untuk menghasilkan batasan tertentu yang memaksa solusi optimum dari masalah LP untuk bergerak mengarah ke pemecahan masalah integer yang diinginkan yaitu :

1. *Branch and bound*
2. Bidang pemotong

Pada kedua metode di atas, batasan akan ditambahkan dan dapat menghilangkan beberapa solusi yang dihasilkan namun tidak pernah menghilangkan solusi integer yang layak. Pada kenyataannya, kedua metode tersebut tidak dapat dinyatakan secara terus menerus efektif dalam memecahkan masalah ILP. Tapi metode *branch and bound* lebih efektif dari segi perhitungan dibanding metode bidang pemotong.

Algoritma Branch And Bound

Untuk lebih memahami algoritma ini, contoh soal berikut akan menerangkan dasar-dasar algoritma *branch and bound*.

Contoh Soal

Sebuah perusahaan gula pasir memproduksi 2 jenis gula pasir dengan kualitas biasa (A) dan premium (B). masing-masing jenis produk melalui tahapan proses yang sama yaitu pengolahan tebu dan pengeringan Kristal gula. Waktu yang diperlukan untuk pembuatan gula A adalah 6 jam dan gula B adalah 5 jam. Sedangkan waktu yang diperlukan untuk pengeringan gula A adalah 2 jam dan untuk gula B adalah 3 jam. Perusahaan tersebut hanya mempunyai waktu untuk pengolahan tebu selama 30 jam dan waktu pengeringan 12 jam perminggu. Gula A menghasilkan keuntungan Rp. 800,00 per kg dan gula B menghasilkan keuntungan Rp. 700,00. Berapa banyak gula yang harus diproduksi agar memperoleh keuntungan maksimal?

Model Matematis

Misal :

x_1 = Gula A

x_2 = Gula B

Keuntungan max : $z = 8x1 + 7x2$

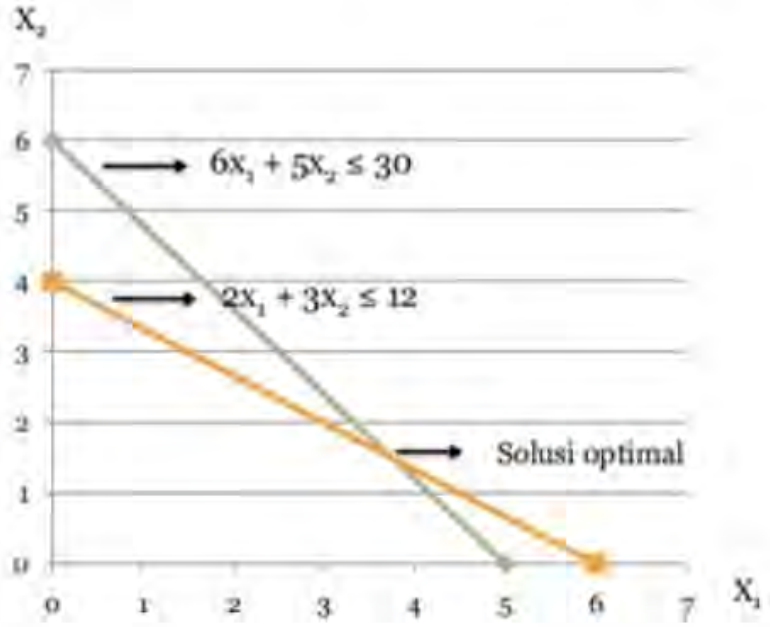
Kendala :

$57 + 5x_2 \leq 30$

$2x_1 + 3x_2 \leq 12$

$x_1, x_2 \geq 0$

Penyelesaian



Solusi Optimal

$X_1 = 3,75$

$X_2 = 1,5$

Keuntungan maksimal

$Z = 8X_1 + 7X_2$

$= 8\ (3,75) + 7\ (1,5)$

$= 40,5$

Bisa bentuk pecahan karena produknya dapat dipecah, bagaiman dengan produk yang diskrit?

Sehingga untuk mendapatkan keuntungan maksimal, pabrik harus menghasilkan 3,75 kg gula A dan 1,5 kg gula B

Contoh lain

Sebuah perusahaan lemari memproduksi 2 jenis lemari, yaitu lemari A dan lemari B. masing-masing produk melalui tahapan proses yaitu persiapan (pemotongan kayu, penghalusan kayu, dan perakitan)) serta penyelesaian (penghalusan,pengecatan, dan pengeringan). Waktu yang diperlukan untuk persiapan lemari A yaitu 6 jam dan lemari B yaitu 5 jam. Sedangkan waktu penyelesaian untuk lemari A adalah 2 jam dan lemari B adalah 3 jam. Perusahaan tersebut hanya mempunyai waktu

untuk pengolahan tebu selama 30 jam dan waktu pengeringan 12 jam perminggu. lemari A menghasilkan keuntungan Rp. 800.000,00 per unit dan lemari B menghasilkan keuntungan Rp. 700.000,00. Berapa banyak lemari yang harus diproduksi agar memperoleh keuntungan maksimal?

Penyelesaian

Dengan menggunakan cara penyelesaian yang sama dengan soal sebelumnya diperoleh pabrik harus menghasilkan 3,75 unit lemari A dan 1,5 unit lemari B. solusi tidak dapat digunakan karena lemari merupakan jenis produk diskrit yang tidak bisa bernilai pecahan

Sehingga cara penyelesaiannya

- Metode round off
- Metode branch and bound
- Mentode gomory (Algoritma pemotongan)

Dengan metode pembulatan langsung (*round off*) dari solusi optimal (x_1=3,75 ; x_2=1,5) diperoleh hasil yang paling mendekati yaitu

X_1 = 4 unit

X_2 = 1 unit

Pada sub bab ini akan dibahas secara rinci metode penyelesaian dengan menggunakan branch and bound. Karena hasil yang diperoleh mengandung variabel yang tidak bulat maka dilakukan pencabangan (*branching*). Jika terdapat variabel yang tidak bulat (misal : xj*), maka dibentuk dua program bilangan bulat baru dengan kendala xj ≤ i1 atau kendala xj ≥ i2. i1 dan i2 adalah dua bilangan bulat tak negatif yang berurutan. Dari soal di atas, diperoleh hasil optimal x_1 =3,75 ; x_2 =1,5 ; dan z = 40,5. Hasil yang diperoleh tidak layak karena bentuknya pecahan sehingga tidak memenuhi persyaratan integer. Algoritma branch and bound memodifikasi ruang pemecahan masalah LP yang pada akhirnya akan memberikan solusi optimum untuk permasalahan ILP. Solusi optimum LP (x_1 = 3,75) tidak memenuhi persyaratan integer sehingga titik ini tidak layak dan dilakukan modifikasi dengan mencari titik lain yang integer pada titk 3 dan 4 ($3 < x_1 > 4$). Ruang pemecahan baru membuat adanya batasan baru yaitu $x_1 \leq 3$ dan $x_1 \geq 4$. Untuk menemukan solusi optimum dengan menggunakan algoritma *branch and bound*, Perhatikan langkah di bawah ini:

Cabang A

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$

$2x1 + 3x_2 \leq 12$ FEASIBLE

$x_1 \leq 3$

$x_1, x_2 \geq 0$, dan integer

Hasil dengan LP sederhana

$x_1 = 3 ; x_2 = 2 ; z = 38$

Cabang B

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$

$2x_1 + 3x_2 \leq 12$ Belum FEASIBLE

$x_1 \geq 4$

$x_1, x_2 \geq 0$, dan integer

Hasil dengan LP sederhana

$x_1 = 4 ; x_2 = 1,2 ; z = 40,4$

Dari cabang B diperoleh hasil $x_1 = 4 ; x_2 = 1,2$; dan $z = 40,4$. Nilai x_2 tidak memenuhi syarat sehingga dilakukan percabangan lagi

Cabang C

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$

$2x_1 + 3x_2 \leq 12$ Belum FEASIBLE

$x_1 \geq 4$

$x_2 \leq 1$

$x_1, x_2 \geq 0$, dan integer

Hasil dengan LP sederhana

$x_1 = 1 ; x_2 = 4,16 ; z = 40,33.$

Cabang D

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$

$2x_1 + 3x_2 \leq 12$ Tidak Layak

$x_1 \geq 4$

$x_2 \geq 2$

$x_1, x_2 \geq 0$, dan integer

Syarat $x_1 \geq 4$ dan $x_2 \geq 2$, di luar area

Dari cabang C diperoleh hasil $x_1 = 1$; $x_2 = 4{,}16$ dan $z = 40{,}33$. Nilai x_2 tidak memenuhi syarat sehingga dilakukan percabangan lagi

Cabang E

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$ FEASIBLE

$2x_1 + 3x_2 \leq 12$

$x_1 \leq 4$

$x_2 \leq 1$

$x_1, x_2 \geq 0$, dan integer

Hasil dengan LP sederhana

$x_1 = 4$; $x_2 = 1$; $z = 39$

Cabang F

Maksimumkan : $Z = 8x_1 + 7x_2$

Kendala :

$6x_1 + 5x_2 \leq 30$ FEASIBLE

$2x_1 + 3x_2 \leq 12$

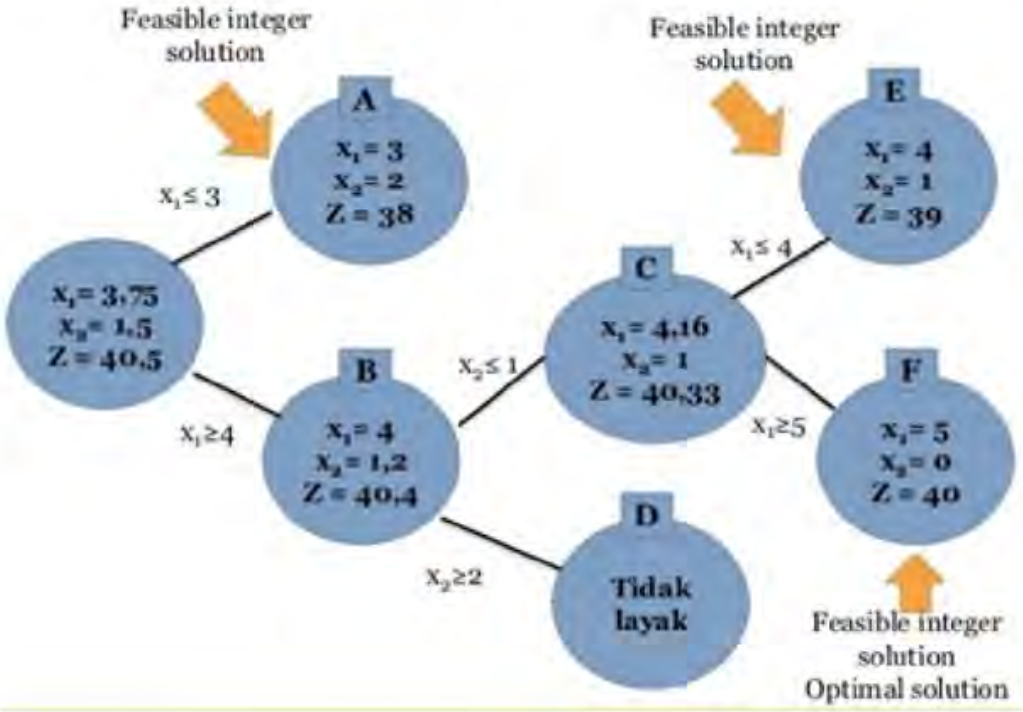
$x_1 \geq 5$

$x2 \geq 2$

$x_1, x2 \geq 0$, dan integer

Hasil dengan LP sederhana

$x_1 = 5$; $x_2 = 0$; $z = 40$



Latihan Soal

Sebuah bengkel dan pencucian mobil menerima layanan untuk 3 jenis mobil yaitu mobil A, B dan C. Biaya perawatan, perbaikan dan pencucian mobil tiap jenis mobil ditabulasikan di bawah.

Mobil	Biaya Perawatan (Ribu)	Biaya Perbaikan (Ribu)	Pencucian (Ribu)
A	200	300	100
B	250	300	85
C	150	200	50

Keuntungan yang diperoleh dari masing-masing layanan yaitu perawatan sebesar Rp. 85.000, perbaikan sebesar Rp. 100.000 dan pencucian sebesar Rp. 25.000. Hitunglah berapa mobil yang datang ke bengkel untuk memperoleh keuntungan maksimal.

Penyelesaian Masalah ILP Dengan R

Berikutnya, kita akan menyelesaikan masalah pada contoh ILP1 menggunakan R. Kode berikut akan digunakan untuk menyelesaikan pemrograman linier integer (*all integer*), diketik di R editor.

```
library(lpSolve)
> #EXAMPLE ILP1
> #MAX Z = 8X1+7X2
> # Constraint
> # 2X1+5X2<=30
> # 2X1+3X2<=12
> # X1, X2 >= 0
> # X1, X2 = Int
#defining parameters
> f.obj<-c(800000,700000)
> f.con<-matrix(c(6,5,2,3),nrow=2,byrow=TRUE)
> f.dir<-c("<=","<=")
> f.rhs<-c(30,12)
#solving model
> lp("max",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)
Success: the objective function is 4e+06
#accessing to R output
> lp("max",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)$solution
[1] 5 0prod.sol$solution #decision variables values
```

Pada bagian *#defining parameters* kita akan mendefinisikan fungsi tujuan dan fungsi kendala.

Fungsi tujuan pada kasus ini adalah *MAX* $z = 800000X1 + 700000X2$, dinyatakan dalam kode R sebagai berikut

```
f.obj<-c(800000,700000)
```

fungsi tujuan ini didefinisikan dalam sebuah vektor bernama f.obj yang berisi angka 800000 dan 70000. Pada kasus ini berarti variabel X1 bernilai 800000 dan X2 bernilai 700000.

Selanjutnya untuk fungsi kendala, dinyatakan dalam kode R sebagai berikut

```
f.con<-matrix(c(6,5,2,3),nrow=2,byrow=TRUE)
```

matriks f.con yang berisi nilai kendala didefinisikan dengan jumlah baris sebanyak 2 (nrow = 2), yang akan membuat matriks ini berdimensi 2x2 (karena diisi perbaris menggunakan byrow = TRUE). Sehingga tampilan matriks kendala adalah sebagai berikut.

```
     [,1] [,2]
[1,]    6    5
[2,]    2    3
```

```
f.dir<-c("<=","<=")
```

menunjukkan tanda pertidaksamaan untuk masing-masing fungsi kendala. Dinyatakan dalam bentuk vektor yang disimpan dalam variabel f.dir

```
f.rhs<-c(30,12)
```

menyatakan vektor yang bernama f. rhs, berisi nilai sisi kanan dari masing-masing fungsi kendala.

Selanjutnya, bagian *#solving model*, seluruh parameter yang akan telah didefinisikan akan dipanggil dengan fungsi lp().

```
lp("max",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)
```

pada kode R di atas, dapat dilihat bahwa ada tulisan "max" yang menyatakan bahwa model ini bertujuan untuk memaksimalkan. Ketika model yang diinginkan bertujuan untuk meminimalkan, maka gunakan "min", akan diperoleh output Success: the objective function is 4e+06

Bagian *#accessing to R output* akan menampilkan ouput dari model pemrograman linier.

```
lp("max",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)$solution
```

akan diperoleh output [1] 5 0

Dapat dilihat bahwa, model ini mempunyai solusi, dinyatakan dengan kata *Success*, sedangkan nilai dari fungsi tujuannya adalah 4e+06. Nilai dari variabel keputusan untuk X1 bernilai 5 dan X2 bernilai 0. Sedangkan jika ingin melihat dual nya dapat menggunakan kode

```
> prod.sol$duals #includes duals of constraints and reduced costs of variables
[1]  0  8 -20  0  0
```

Berikutnya, kita akan menyelesaikan masalah pada contoh ILP menggunakan R. Kode berikut akan digunakan untuk menyelesaikan pemrograman linier integer (**biner**), diketik di R editor.

```
library(lpSolve)
#EXAMPLE ILP BIN
> #MAX Z = 15X1+18X2+9X3+8X4+8X5
> # Constraint
> #10X1+14X2+16X3+8X4+9X5<=50
> #8X1+11X2+12X3+9X4+7X5<=40
> #6X1+8X2+9X3+7X4+5X5<=30
#defining parameters
> f.obj<-c(15,18,9,8,8)
> f.con< matrix(c(10,14,16,8,9,8,11,12,9,7,6,8,9,7,5),nrow=3,byrow=
TRUE)
> f.dir<-c("<=","<=","<=")
> f.rhs<-c(50,40,30)
#solving model
> lp("max",f.obj.f.con,f.dir,f.rhs,all.bin=TRUE)
Error in is.data.frame(objective.in) : object 'f.obj.f.con' not found
> lp("max",f.obj,f.con,f.dir,f.rhs,all.bin=TRUE)
Success: the objective function is 50
#accessing to R output
> lp("max",f.obj,f.con,f.dir,f.rhs,all.bin=TRUE)$solution
[1] 1 1 1 1 0
```

Pada bagian *#defining parameters* kita akan mendefinisikan fungsi tujuan dan fungsi kendala.

Fungsi tujuan pada kasus ini adalah *MAX* z = 15X1 + 18X2 + 9X3 + 8X4 + 8X5, dinyatakan dalam kode R sebagai berikut

```
f.obj<-c(15,18,9,8,8)
```

fungsi tujuan ini didefinisikan dalam sebuah vektor bernama f.obj yang berisi angka 15,8,9,8,dan 8. Pada kasus ini berarti variabel X1 bernilai 15, X2 bernilai 18, X3 bernilai 9, X4 bernilai 8, dan X% bernilai 8.

Selanjutnya untuk fungsi kendala, dinyatakan dalam kode R sebagai berikut

```
f.con< matrix(c(10,14,16,8,9,8,11,12,9,7,6,8,9,7,5),
nrow=3,byrow=TRUE)
```

matriks f.con yang berisi nilai kendala didefinisikan dengan jumlah baris sebanyak 3 (nrow = 3), yang akan membuat matriks ini berdimensi 5x3 (karena diisi perbaris menggunakan byrow = TRUE). Sehingga tampilan matriks kendala adalah sebagai berikut.

```
     [,1] [,2] [,3] [,4] [,5]
[1,]   10   14   16    8    9
[2,]    8   11   12    9    7
[3,]    6    8    9    7    5
```

```
f.dir<-c("<=","<=","<=")
```

menunjukkan tanda pertidaksamaan untuk masing-masing fungsi kendala. Dinyatakan dalam bentuk vektor yang disimpan dalam variabel f.dir

```
f.rhs<-c(50,40,30)
```

menyatakan vektor yang bernama f. rhs, berisi nilai sisi kanan dari masing-masing fungsi kendala.

Selanjutnya, bagian *#solving model*, seluruh parameter yang akan telah didefinisikan akan dipanggil dengan fungsi lp().

```
> lp("max",f.obj,f.con,f.dir,f.rhs,all.bin=TRUE)
```

pada kode R di atas, dapat dilihat bahwa ada tulisan "max" yang menyatakan bahwa model ini bertujuan untuk memaksimalkan. Ketika model yang diinginkan bertujuan untuk meminimalkan, maka gunakan "max", akan diperoleh output Success: the objective function is 50

Bagian *#accessing to R output* akan menampilkan ouput dari model pemrograman linier.

```
lp("max",f.obj,f.con,f.dir,f.rhs,all.bin=TRUE)$solution
```

akan diperoleh output [1] 1 1 1 1 0

Dapat dilihat bahwa, model ini mempunyai solusi, dinyatakan dengan kata *Success*, sedangkan nilai dari fungsi tujuannya adalah 50. Nilai dari variabel keputusan untuk X1 bernilai 1, X2 bernilai 1, X3 bernilai 1, X4 bernilai 1, dan X5 bernilai 0.

BAB 6
MODEL TRANSPORTASI

Pengantar Pemrograman Linier : Model Transportasi
Model Matematis Transportasi
Pemecahan Awal Problem Transportasi
Metode Optimalisasi Problem Transportasi Lainnya
Penyimpangan Dalam Problem Transportasi
Penyelesaian Masalah Transportasi Dengan R

MODEL TRANSPORTASI

Pengantar Pemrograman Linier : Model Transportasi

Pada bab ini membahas penggunaan program linier dalam masalah transportasi atau biasa dikenal dengan sebutan model transportasi. Model transportasi berkaitan dengan perencanaan pengiriman produk dari beberapa sources atau sumber (contohnya tempat produksi atau distributor) ke beberapa destinasi atau tujuan (misalnya *warehouse* atau ritel) dengan biaya yang paling rendah. Model transportasi dapat disesuaikan dengan kebutuhan untuk misalnya penjadwalan dan penugasan tenaga kerja.

Pada dasarnya model transportasi ini merupakan turunan atau pengembangan dari programma linier sehingga sebenarnya bisa saja diselesaikan menggunakan metode simpleks biasa. Perbedaannya ada pada struktur dan kompleksitas variabel yang terlibat yang memerlukan sebuah pengembangan sebuah algoritma atau *procedur* pemecahan khusus yang biasa disebut teknik/metode transportasi yang memungkinkan pengembangan sebuah prosedur pemecahan yang disebut teknik transportasi yang proses perhitungannya lebih cepat, efektif dan efisien dibanding menggunakan pendekatan programa linier umumnya.

Teknik/metode transportasi merupakan salah satu pendekatan bisa dipakai untuk membantu membuat keputusan dalam mengatur alokasi distribusi dari berbagai sumber *supply* (*supply source*) yang memproduksi/menyediakan barang/produk ke lokasi produk tersebut dibutuhkan secara optimal sehingga total cost dari distribusi alokasi tersebut minimal. Teknik transportasi terkait dengan alokasi distribusi *single product* atau satu produk dari sejumlah sumber *supply* (penawaran) yang dibatasi kapasitas *supply*, menuju ke sejumlah destinasi (tujuan) dengan sejumlah permintaan/kebutuhan tertentu dengan biaya pengiriman (distribusi) yang minimal (Simbolon, Situmorang, & Napitupulu, 2014). Biaya minimum dapat diperoleh dengan mengalokasikan produk sedemikian rupa karena terdapat perbedaan biaya alokasi untuk tiap lokasi yang berbeda pula, baik dari sumber ke tujuan dan sebaliknya.

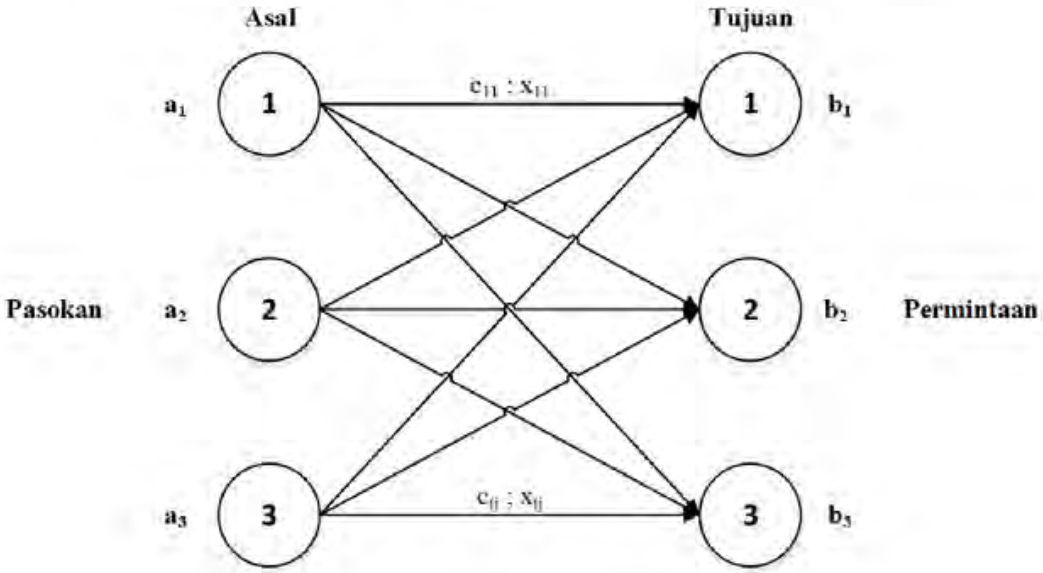
Model transportasi dapat digunakan untuk memecahkan berbagai masalah seperti permasalahan distribusi (lokasi-alokasi), permasalahan bisnis yang mencakup alokasi investasi modal, analisis lokasi, perencanaan produksi, dan sebagainya. Dalam bab ini akan dibahas penggunaan model transportasi dalam distribusi produk. Pada dasarnya, metode transportasi mencoba menentukan sebuah planning distribusi produk tunggal dari beberapa sumber produksi/gudang ke beberapa tujuan permintaan yang mencakup beberapa data yang dibutuhkan menurut (Taha, 1996) seperti berikut ini:

a. Kapasitas atau tingkat penawaran di setiap sumber produksi atau sumber *supply* dan tingkat permintaan (*demand*) pada tiap destinasi tujuan.
b. *Unit transportation cost* atau biaya transportasi setiap unit produk dari setiap sumber produksi/pengiriman ke setiap lokasi/destinasi permintaan.

Akibat hanya terdapat satu jenis barang, sehingga sebuah tujuan dapat menerima permintaannya lebih dari satu sumber. Model ini bertujuan untuk menentukan jumlah barang yang harus dikirimkan ke setiap tujuan dengan biaya transportasi total yang minimum.

Model Matematis Transportasi

Seperti yang sudah dibahas sebelumnya, model transportasi bertujuan untuk memperoleh biaya transportasi yang minimum sehingga dapat diasumsikan bahwa biaya per unit produk yang dikirim pada jalur distribusi tertentu sebanding atau proporsioanl secara langsung dengan berapa jumlah produk yang didistribusikan dari rute tersebut. Gambar di bawah ini menunjukan sebuah model transportasi yang berisi sebuah *network* atau jaringan dengan m sumber *supply* (asal) dan n destinasi permintaan (tujuan). **Node** mewakili sumber penawaran dan destinasi/ tujuan serta **busur** mewakili rute pengiriman barang dari sumber ke tujuan. ai menunjukan jumlah penawaran di sumber i, bj mewakili jumlah permintaan di tujuan j, serta cij adalah biaya unit transportasi antara sumber i ke j dan xij mewakili jumlah barang yang didistribusikan dari asal sumber i ke tujuan j.



Gambar 6. 1 Model Transportasi

Model LP yang mewakili masalah transportasi yaitu :

ai = jumlah pasokan pada sumber (asal) i

bi = jumlah permintaan pada tujuan j

ci = biaya transportasi dari sumber i ke tujuan j

Sehingga dari network di atas, maka beriktunya kita bisa membuat formulasi linier programming sebagai berikut:

Minimumkan

$$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \; x_{ij}$$

Dengan batasan

$$\sum_{j=1}^{n} x_{ij} \le a_i , \qquad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} \le b_j , \qquad i = 1, 2, \ldots, m$$

$$x_{ij} \ge 0 , \qquad \text{for all i dan j}$$

Dari sistem persamaan LP di atas, kita membuat definisi persamaan dari masing-masing formulasi. Pada persamaan kendala yang pertama diperuntukkan untuk menetapkan atau mengkondisikan bahwa jumlah yang dikirimkan dari sebuah sumber penawaran (asal) tidak boleh melebihi kapasitas sumber penawarannya. Sementara itu, pada kendala yang kedua mengkondisikan bahwa jumlah yang akan dikirim ke sebuah tujuan permintaan harus sesuai atau memenuhi jumlah permintaannya. Dalam bentuk tabel, model transportasi dapat disajikan seperti pada Gambar 2.

		Tujuan				
		1	2	...	n	Persediaan
Sumber	1	c 11 x11	c 11 x12	...	c 1n x1n	a1
	2	c 21 x21	c 22 x22	...	c 2n x2n	a2
	...	... ...	... ...	... ...	... ...	...
	m	c m1 xm1	c m2 xm2	...	c mn xmn	am
Permintaan		b1	b2	...	bn	

Gambar 6. 2 Tabel Transportasi

Ketika total penawaran sama dengan total permintaan, maka model transportasi yang dihasilkan disebut model transportasi berimbang yang diwakili oleh persamaan berikut :

$$\sum_{j=1}^{m} a_i = \sum_{i=1}^{n} b_i$$

Dengan batasan

$$\sum_{j=1}^{n} x_{ij} = a_i \,, \qquad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \,, \qquad i = 1, 2, \ldots, m$$

Jika terjadi kondisi dimana pada kendala tidak terpenuhi atau dengan kata lain jumlah *supply* atau penawaran lebih besar atau bahkan lebih kecil dari jumlah yang diminta di sisi perminataan tujuan, maka bisa dipastikan kondisi tersebut akan menyebabkan ketidakseimbangan model jaringan transportasi. Sehingga diperlukan perlakuan khusus dari model jaringan transportasi tersebut agar bisa diselesaikan, yaitu dengan cara menyeimbangkan penawaran dan permintaan dengan memberi tambahan variabel baru yang bisa dikenal sebagai variabel *dummy* agar jaringan permintaan dan penawaran menjadi seimbang dan bisa diselesaikan dengan metode transportasi.

Contoh 1 (Model Transportasi Standar).

DG Auto Mobile memiliki pabrik di kota A, B dan C. Pusat terletak di kota D dan E. Diketahui bahwa ketiga tempat produksi tersebut mempunyai kapasitas produksi yang berbeda sebesar 1.000, 1.500 dan 1.200 unit mobil setiap periode tahunnya. Sementara itu, terdapat dua *distribution centre* atau pusat distribusi mempunyai permintaan sejumlah 2.300 dan 1.400 unit mobil per tahunnya. Biaya kirim atau biya transportasi setiap unit mobil per kilometer jarak kirim diketahui seb[illegible] 8 dolar. Sedangkan data jarak tempuh (dalam satuan kilometer) dari

masing-masing lokasi pabrik ke masing-masing lokasi pusat distribusi dapat dilihat di bagan di bawah berikut ini:

	D	E
A	1000	2690
B	1250	1350
C	1275	850

Dari tabel jarak tersebut bisa diartikan bahwa jarak yang diperlukan untuk mengirim sejumlah unit mobil dari pabrik A ke pusat distribusi D sebesar 1.000 kilometer dan jarak dari pabrik lain ke tujuan lain bisa didefinisikan dengan cara yang sama. Kemudian tabel jarak di atas kemudian dikonversikan ke dalam biaya transportasi dari pabrik ke lokasi tujuan dengan mengalikannya dengan biaya kirim per kilometernya (8 dolar per unit mobil per kilometer). Sehingga tabel biaya kirim (satuan puluhan) dari masing-masing pebrik ke lokasi pusat distribusi yang mewakili cij dalam model umum bisa dilihat di bawah ini:

		D	E
	A		
(1)		80	215
	B		
(1)		100	108
	C		
(1)		102	68

Dari tabel di atas dapat diketahui bahwa model transportasi yang dihasilkan berimbang karena total penawaran (= 1000 + 1500 + 1200 = 3700) sama dengan total permintaan (= 2300 + 1400 = 3700). Model LP yang dihasilkan memiliki batasan yang semuanya berbentuk persamaan:

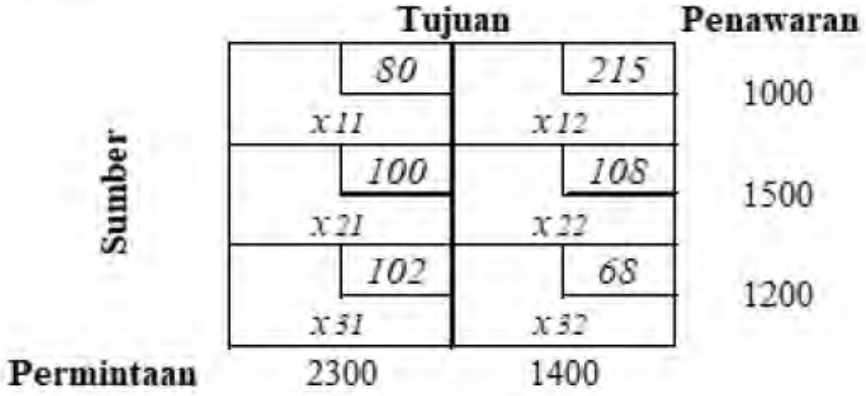
Minimumkan $z = 80x11 + 215x12 + 100x21 + 108x22 + 102x31 + 68x32$

Dengan batasan

$$x_{11} + x_{12} = 1000$$
$$+ x_{21} + x_{22} = 1500$$
$$+ x_{31} + x_{32} = 1200$$
$$x_{11} + x_{21} + x_{31} = 2300$$
$$+ x_{12} + x_{22} + x_{32} = 1400$$

$xij \geq 0$, untuk semua i dan j

Sebuah metode yang lebih ringkas untuk mewakili model transportasi disebut tabel transportasi. Tabel ini berbentuk matriks dengan baris yang mewakili sumber dan kolom mewakili tujuan. Unsur biaya c_{ij} berada di pojok kanan atas setiap sel (i,j). Model dapat diringkas seperti gambar di bawah ini.



Pemecahan Awal Problem Transportasi

Pembahasan pada sub bab ini yaitu memecahkan permasalahan transportasi dengan menggunakan teknik transportasi. Metode ini menggunakan langkah-langkah metode simpleks secara langsung dengan sedikit perbedaan pada perincian penerapan kondisi optimalisasi dan kelayakan.

Langkah-langkah dasar dari teknik transportasi adalah sebagai berikut :

1. Langkah 1 : menentukan solusi pertama (*initial solution*) yang *feasible* atau layak

2. Langkah 2 : menentukan *incoming* variabel atau variabel yang dimasukkan sebagai pengganti variabel yang keluar di antara variabel non dasar. Seperti pada metode simpleks, jika semua variabel masuk memenuhi kondisi optimalisasi, maka hentikan proses; atau jika tidak, maka silahkan dilanjut ke langkah berikutnya (langkah ketiga).
3. Langkah 3 : menentukan *outcoming variable* atau variable yang dikeluarkan (berdasarkan kondisi layak) di antara variabel-variabel basis yang ada, kemudian temukan pemecahan basis yang baru. Selanjutnya kembali lanjutkan ke langkah kedua.

Karena *initial step* merupakan langkah awal yang sangat krusial untuk langkah-langkah selanjutnya, maka ketelitian dan kehati-hatian dalam memasukkan koefisien dan variabel yang diperlukan. Untuk membuat dan menentukan *initial solution* di step awal yang *feasible*, terdapat sejumlah cara/pendekatan yang bisa dipakai antara lain memakai pendekatan berdasarkan biaya terendah (*least-cost*) dan metode sudut barat laut (*northwest-corner*).

a. Biaya Terendah (Least Cost)

Pada dasarnya cara kerja pendekatan *least cost* adalah dengan memberikan keutamaan atau prioritas alokasi pada sel di antara sumber dan permintaan yang mempunyai biaya distribusi/kirim per satuan unit dan jarak yang terkecil (termurah). Proses alokasi awal (*initial allocation*) dilakukan pada sel atau kotak tabel transportasi yang memiliki biaya kirim termurah.

Tahapan proses penyelesaian problem transportasi dengan menggunakan pendekatan *least cost* adalah sebagai berikut:

1. Tentukan sel atau kotak dari tabel transportasi yang punya biaya transportasi (c_{ij}) termurah, kemudian lakukan pengalokasian sejumlah unit sesuai dengan jumlah penawaran dari sumber ke permintaan tujuan semaksimal mungkin yang bisa dilakukan hingga menghabiskan salah satu dari kolom perminatan (kolom j) atau baris penawaran (baris i). Berikutnya baris permintaan (baris i) atau kolom penawaran (kolom j) yang telah terpenuhi seluruhnya (untuk permintaan) atau habis (untuk penawaran) dibuang/dihilangkan dari proses perhitungan selanjutnya.

2. Untuk sel/kotak yang belum habis (terisi), pilih lagi c_{ij} termurah selanjutnya dan alokasikan semaksimal mungkin pada baris i atau kolom j seperti yang dilakukan di langkah 1.
3. Pada langkah selanjutnya, langkah tersebut di atas diulangi lagi ke biaya kirim termurah selanjutnya sampai semua penawaran dan permintaan terpenuhis esuai jumlahnya.

Contoh Soal

Tiga gudang penyimpanan produk dengan kapasitas masing-masing yaitu 90 ton, 60 ton dan 50 ton akan mengirim barang ke tiga kota yang berbeda dengan kebutuhan masing-masing kota yaitu 50 ton, 110 ton serta 40 ton. Biaya pengiriman (ribuan) dari gudang ke kota disajikan dalam tabel berikut.

Tabel 6. 1 Biaya Krim

Gudang	Kota		
	A	B	C
1	20	5	8
2	15	20	10
3	25	10	19

Kita akan mencoba menyelesaikan problem transportasi yang ditulis oleh Yulianto (2012) di atas dengan pnyelesaian awal yang *feasible* dengan memakai pendekatan *least cost* atau biaya termurah.

Penyelesaian

Biaya terkecil yaitu pengiriman dari gudang 1 ke kota B dengan $c_{12} = 5$, sehingga sel ini diisi semaksimal mungkin, yaitu sebesar $x_{12} = 90$. Gudang 1 sudah kehabisan barang sehingga x_{11} dan x_{13} tidak bisa terisi lagi.

Tabel 6. 2 Alokasi Biaya Paling Kecil Awal

		Kota			
		A	B	C	Persediaan
Gudang	1	20	5 90	8	90
	2	15	20	10	60
	3	25	10	19	50
Permintaan		50	110	40	

Dari sisa sel yang masih bisa diisi pengiriman dengan biaya terendah adalah dari pabrik 3 ke kota B dengan biaya $c_{32} = 10$. c_{32} diisi sebanyak 20 untuk memenuhi permintaan kota B yaitu 110 ton.

Dengan cara yang sama, lakukan untuk sel lainnya yang belum terarsir. Penyelesaian fisibel awal dengan metode least cost adalah : biaya total pendistribusian sebesar 90(5) + 20(15) + 40(10) + 30(25) + 20(10) = 2.100 (ribuan).

Tabel 6. 3 Alokasi Biaya Terkecil Berikutnya

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20	5 90	8	90
2	15 20	20	10 40	60
3	25 30	10 20	19	50
Permintaan	50	110	40	

b. North West Corner (NWC)

Metode NWC ini merupakan salah satu pendekatan transportasi yang relatif paling mudah dipakai, namun kembali seperti metode-metode lain, hasil dari penggunaan pendekatan ini juga masih belum tentu ottimal langsung. Pada metode ini, sumber dan tujuan disajikan dalam bentuk matriks (tabel) yang diurutkan dari sebelah/bagian kiri ke arah kanan dan dari arah atas ke bagian bawah. Ada sejumlah aturan yang harus diperhatikan dan diikuti pada pendekatan ini yaitu sebagai berikut :

1. Aturan pertama yang perlu diingat adalah menghabiskan penawaran di setiap baris sebelum akhirnya pindah ke baris setelahnya yang berada di posisi bawahnya
2. Aturan kedua adalah memenuhi syarat jumlah demand/permintaan di tiap kolomnya sebelum melanjutkan pindah ke kolom berikutnya yang berada disebelah kanannya.
3. Aturan selanjutnya adalah memonitor semua nilai penawaran dan permintaan/*demand* apakah sudah sesuai dan tidak ada penawaran yang tersisa atau tidak ada permintaan yang belum dipenuhi seluruhnya.

Contoh Soal

Dengan soal yang sama seperti sebelumnya, tentukan penyelesaian fisibel awal dengan metode north west corner (NWC).

Penyelesaian

Jumlah kapasitas yang dimiliki gudang 1, 2 dan 3 adalah 90 + 60 + 50 = 200 ton, sedangkan jumlah permintaan di setiap kota A, B dan C adalah 50 + 110 + 40 = 200 ton. Jumlah permintaan dan penawaran sama sehingga proses iterasi dapat dimulai. Biaya pengiriman per unit barang ditunjukkan pada ujung kanan atas tiap sel. Di sisi kanan merupakan jumlah persediaan barang dari setiap gudang, sedangkan sisi bawah tabel merupakan jumlah permintaan tiap kota.

Tabel 6. 4 Awal Transportasi

		Kota A	Kota B	Kota C	Persediaan
Gudang	1	20	5	8	90
	2	15	20	10	60
	3	25	10	19	50
Permintaan		50	110	40	

Sisi paling barat laut dari tabel adalah sel $c_{11} = 20$. Sel ini diisi sebanyak mungkin. Gudang 1 mempunyai 90 ton barang, sedangkan kota A memerlukan 50 ton. Karena jumlah permintaan lebih kecil dibanding barang yang dimiliki gudang sehingga x11 diisi sebanyak mungkin yaitu 50 ton sesuai permintaan yang dimiliki kota A. Dengan mengisi $x_{11} = 50$, maka permintaan kota A terpenuhi sehingga x_{12} dan x_{31} tidak boleh diisi lagi.

Tabel 6. 5 Alokasi Awal Pojok Kiri Atas (North West Corner)

		Kota A	Kota B	Kota C	Persediaan
Gudang	1	20 **50**	5	8	90
	2	15	20	10	60
	3	25	10	19	50
Permintaan		50	110	40	

Selanjutnya sel c_{12} =5 akan diisi dengan barang sebanyak mungkin. Gudang 1 hanya mempunyai 90 ton dan sudah dikirimkan ke kota A sebanyak 50 ton sehingga tersisa 40 ton. Di sisi lain, kota B membutuhkan sebanyak 110 ton, maka x_{12} = 40. Gudang 1 sudah kehabisan barang sehingga x_{13} tidak boleh diisi lagi.

Tabel 6. 6 Alokasi Selanjutnya Pada NWC

		Kota A	Kota B	Kota C	Persediaan
Gudang	1	20 **50**	5 **40**	8	90
	2	15	20	10	60
	3	25	10	19	50
Permintaan		50	110	40	

Akibat barang dari gudang 1 sudah habis, selanjutnya ujung barang lautnya adalah sel c_{22} = 20. Gudang 2 mempunyai 60 ton barang, sedangkan kota B tinggal membutuhkan 70 ton barang lagi. Maka x_{22} = 60 dan x_{23} tidak boleh diisi lagi. Sehibgga semua barang sudah tersalurkan.

Tabel 6. 7 Soluasi Awal Transportasi Dengan NWC

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20 (50)	5 (40)	8	90
2	15	20 (60)	10	60
3	25	10 (10)	19 (40)	50
Permintaan	50	110	40	

Penyelesaian fisibel awal dengan metode northwest corner adalah sebagai berikut :

Biata total pendistribusian sebesar 50(20) + 40(5) + 60(20) + 10(10) + 40(19) = 3.260 (ribuan).

Jumlah sel basis (sel terisia) = 5[42]l, yaitu sama dengan jumlah baris + jumlah kolom-1 = 5. Sehingga jumlah basisnya mencukupi dan tidak perlu basis dummy.

Metode Optimalisasi Problem Transportasi Lainnya

Metode yang disajikan sebelumnya belum tentu memberikan solusi yang optimal. Sehingga pada bagian ini menyajikan solusi optimal setelah dilakukan pemecahan awal dengan met[2]e sebelumnya. Sub bab ini akan membahas 2 metode untuk optimalisasi yaitu metode Stepping Stone dan metode Modified Distribution Method (MODI).

a. Stepping Stone

Pendekatan stepping stone ini pada dasarnya adalah digunakan untuk melakukan perubahan alokasi produk atau *re-allocation* untuk mendapatkan alokasi produk yang lebih murah total biaya kirimnya. Pendekatan *stepping stone* ini lebih dkelan dengan pendekatan coba-coba (*trial error*), meskipun pada kenyataannya pendekatan ini tetap memerlukan prosedur atau aturan-aturan tertentu yang cukup konsisten. [23]uran-aturan tarsebut dibuat dengan tujuan untuk memonitor pengurangan biaya per unit yang lebih besar daripada penambahan-penambahan biaya per unitnya jika dilakukan re-alo[28]i. Untuk menyelesaikan kasus dengan menggunakan metode ini, jumlah rute

atau sel yang mendapat alokasi harus sebanyak (jumlah kolom + jumlah baris) – 1. Berikut langkah pengerjaan dengan metode *stepping stone* :

1. Lakukan pemilihan salah satu sel yang kosong atau tidak mendapat alokasi atau biasa kita sebut sebagai sel non basis karena tidak berisi alokasi sama sekali.
2. Dimulai dari sel non basis ini, buat jalur tertutup atau close path melalui sel yang terdapat alokasi (biasa disebut sel basis karena beralokasi) menuju sel kosong terpilih. Jalur kosong ini hanya boleh bergerak secara horizontal dan vertical saja.
3. Berikutnya, kita beri tanda positif (+) pada sel kosong terpilih kemudian kita beri tanda negative (-) dan (+) secara bergantian pada setiap sudut jalur tertutup.
4. Langkah selanjutnya, kita lakukan perhitungan nilai (indeks) perbaikan dengan cara menjumlahkan biaya transportasi pada sel bertanda (+) dan mengurangkan biaya transportasi pada sel bertanda (-) yang sudah di langkah sebelumnya.
5. Tahap paling akhir adalah mengulangi tahap pertama sampai dengan tahap empat sampai diperoleh sebuah kondisi dimana semua nilai (indeks) perbaikan untuk seluruh sel non basis (sel kosong). Apabila terajadi sebuah situasi dimana indeks perbaikan dari sel non basis yang ada lebih besar atau sama dengan nol (0), maka solusi optimal terlah tercapai dan tidak perlu dilanjutkan ke tahap selanjutnya.

Contoh Soal

Tiga gudang penyimpanan produk dengan kapasitas masing-masing yaitu 90 ton, 60 ton dan 50 ton akan mengirim barang ke tiga kota yang berbeda dengan kebutuhan masing-masing kota yaitu 50 ton, 110 ton dan 40 ton. Biaya pengiriman (ribuan) dari gudang ke kota disajikan dalam tabel berikut.

Tabel 6. 8 Biaya Kirim ($/unit)

Pabrik	Kota		
	A	B	C
1	20	5	8
2	15	20	10
3	25	10	19

Masih menngguna6n soal yang sama yang diberikan oleh Yulianto (2012), kita coba hitung total biaya optimal yang harus dikeluarkan oleh perusahaan untuk memenuhi permintaan ketiga kota tersebut dengan pendekatan *stepping stone*.

Penyelesaian

1. Tabel Transportasi

Tabel 6. 9 Awal Transportasi

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20	5	8	90
2	15	20	10	60
3	25	10	19	50
Permintaan	50	110	40	200

2. Dengan metode north west corner, tabel fisibel awal yaitu sebagai berikut

Tabel 6. 10 Solusi Awal Dengan NWC

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20 **50**	5 **40**	8	90
2	15	20 **60**	10	60
3	25	10 **10**	19 **40**	50
Permintaan	50	110	40	200

3. Tabel alokasi pertama dengan metode stepping stone

Tabel 6. 11 Alokasi Pertama Stepping Stone

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20 (50)	5 (40)	8	90
2	15	20 (60)	10	60
3	25	10 (10)	19 (40)	50
Permintaan	50	110	40	200

Biaya mengiriman untuk alokasi tahap pertama :

$50(20) + 40(5) + 60(20) + 10(10) + 40(19) = 3260$

4. Menguji sel-sel yang masih kosong, apakah masih bisa memiliki nilai negatif atau tidak. Artinya masih bisa menurunkan biaya transportasi atau tidak. Sel yang diuji yaitu sel $x_{13}, x_{21}, x_{23}, x_{31}$. Pengujian dilakukan pada setiap sel yang kosong dengan menggunakan metode *stepping stone*. Pengujian dilakukan mulai dari sel kosong tersebut, kemudian bergerak secara lurus/tidak boleh diagonal (boleh searah jarum jam atau berlawanan arah jarum jam) ke arah sel yang telah terisi dengan alokasi. Lakukan hal seperti itu terus sampai kembali ke sel yang kosong. Setiap pergerakan akan mengurangi atau menambah biaya secara bergantian pada sel kosong tersebut. Perhatikan tanda panah dan tanda (+) atau (-) nya.

Tabel 6. 12 Hasil Uji Sel-Sel Non Basis

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20 (50)	5 (40)	8	90
2	15	20 (60)	10	60
3	25	10 (10)	19 (40)	50
Permintaan	50	110	40	200

Pengujian:

Sel $x_{13} = 8 - 19 + 10 - 6 = -6$

Sel $x_{21} = 15 - 20 + 5 - 20 = -20$

Sel $x_{23} = 10 - 19 + 10 - 20 = -19$

Sel $x_{31} = 25 - 20 + 5 - 10 = 0$

5. Merubah alokasi pengiriman ke sel x_{21}, yang pengujian sebelumnya memiliki pergerakan:

Tabel 6. 13 Perubahan Alokasi

Gudang	Kota A	Kota B	Kota C	Persediaan
1	20 (-)	5 (+)	8	90
2	15 (+)	20 (-)	10	60
3	25	10	19	50
Permintaan	50	110	40	200

Dari pergerakan dan tanda (+)/(-) yang ada, perhatikan sel yang bertanda minus saja yaitu sel x_{11} dan x_{12}. Di antara dua sel yang bertanda minus, pilih sel yang alokasi pengiriman sebelumnya paling kecil dan yang terpilih adalah x_{11} dengan alokasi sebelumnya 50 ton dan lebih kecil dari alokasi sel x_{22} yang 60 ton.

6. Selanjutnya, angka 50 ton di sel terpilih digunakan untuk mengurangi atau menambah alokasi yang ada selama pengujian. Sehingga diperoleh hasil sesuai tabel berikut:

Tabel 6. 14 Re-Alokasi Hasil Perhitungan

Gudang	Kota A	Kota B	Kota C	Kapasitas
1	20	5 **90**	8	90
2	15 **50**	20 **10**	10	60
3	25	10 10	19 40	50
Permintaan	50	110	40	200

Sel x_{11} menjadi 0 karena 50 – 50 = 0

Sel x_{12} menjadi 90 karena 40 + 50 = 90

Sel x_{22} menjadi 10 karena 60 – 10 = 10

Sel x_{21} menjadi 50 karena 0 + 50 = 50

Pengujian :

Sel x_{11} = 20 – 5 + 20 – 15 = 20 (menjadi lebih mahal 20/ton)

Sel x_{13} = 8 – 19 + 10 – 5 = -6

Sel x_{23} = 10 – 19 + 10 – 20 = -19

Sel x_{31} = 25 – 15 + 20 – 10 = 20 (menjadi lebih mahal 20/ton)

Dari hasil pengujian tersebut, ternyata sel x_{23} masih dapat memberikan penurunan biaya sebesar Rp 19/ton. Dengan demikian, perubahan alokasi perlu dilakukan dengan mencoba mengalokasikan pengiriman ke sel x_{23} dengan langkah yang sama dengan sebelumnya.

b. Modified Distribution (MODI)

Pendekatan lain selain *stepping stone* dalam optimisasi metode transportasi adalah dengan menggunakan pendekatan MODI (*modified distribution*). MODI yang merupakan pengembangan dari pendekatan *stepping stone*, menggunakan pendekatan perbaikan indeks atau nilai baris dan kolom dalam proses atau algoritmanya. Sebelum memahami prosedur dan langkah-langkah penyelesaian dengan pendekatan MODI perlu dipahami beberapa istilah dan dasar yang akan digunakan dalam algoritma penyelesaiannya. Misalnya istilah sel (bilik) basis dan sel non basis. Sel basis dipakai untuk mendefiniskan sel yang ada alokasi dari sumber ke tujuan sedangkan sel non basis adalah sel yang yang tidak alokasi dari sumber ke tujuan alias sel kosong. Salah satu kelebihan MODI dibanding pendekatan lainnya adalah dalam hal penentuan sel-sel kosong yang bertujuan untuk menentukan penghematan biaya. Denagn MODI, prosedur yang dipakai lebih pasti dan ketepatannya cukup tinggi dibanding *stepping stone* sehingga proses optimalisasi problem lebih cepat ditemukan. Sedangkan hambatan terbesar pemakaian algoritma MODI ini dibanding *stepping stone* adalah proses yang digunakan lebih panjang (karena lebih sistematis).

Adapun formulasi dasar yang digunakan dalam MODI ini adalah dibedakan berdasarkan jenis sel-nya (basis atau non basis) dengan langkah-langkah algoritma sebagai berikut:

Langkah Pertama

Langkah pertama MODI adalah menentukan nilai untuk setiap baris (Ri) dan setiap kolom (Cj) di dalam tabel transportasi. Nilai baris dan nilai kolom ini ditentukan dengan menggunakan koefisien biaya (Bij) yang nilainya dalam tiap sel basis sama dengan nilai baris dan kolom yang salin berkorespondensi atau ri + cj = Bij untuk tiap sel basisnya. Sebagai contoh soal sebelumnya yang sudah dilakukan inisiasi solusi dengan NWC sebagai berikut:

Tabel 6. 15 Alokasi Awal Dengan NWC

		Kota A	Kota B	Kota C	Persediaan
Gudang	1	20 50	5 40	8	90
	2	15	20 60	10	60
	3	25	10 10	19 40	50
Permintaan		50	110	40	200

Dari tabel di atas bisa dilihat sel-sel basis yang telah terisi sejumlah alokasi dari gudang ke kota tujuan dengan alokasi detail sebagai berikut:

Dari gudang 1 ke tujuan A = 50 unit
Dari gudang 1 ke tujuan B = 40 unit
Dari gudang 2 ke tujuan B = 60 unit
Dari gudang 3 ke tujuan B = 10 unt
Dari gudang 3 ke tujuan C = 40 unit

Total biaya yang harus dikeluarkan dengan alokasi tersebut sebesar: $ 3.800

Pada langkah pertama ini tabel alokasi awal di atas kemudian diberi keterangan nilai baris dan kolom sebagai berikut:

Tabel 6. 16 Penentuan Baris Dan Kolom

		$R_1 = ?$	$R_2 = ?$ Kota tujuan	$R_3 = ?$	
		A	B	C	Persediaan
$C_1 = ?$ Gudang	1	20 / 50	5 / 40	8	90
$C1_2 = ?$	2	15	20 / 60	10	60
$C_3 = ?$	3	25	10 / 10	19 / 40	50
Permintaan		50	110	40	200

Berdasarkan tabel di atas, nilai baris (Ri) dan nilai tiap kolom (Cj) dapat ditentukan mempertimbangkan sel-sel basis (sel yang beralokasi), dimana terdapat 5 sel basis yang bisa dipakai untuk membentuk persamaan yang memenuhi persyaratan Ri+Cj = Bij sebagai berikut:

1. Gudang 1 ke kota tujuan A ------> R1 + C1 = 20
2. Gudang 1 ke kota tujuan B ------> R2 + C1 = 5
3. Gudang 2 ke kota tujuan B ------> R2 + C2 = 20
4. Gudang 3 ke kota tujuan B ------> R2 + C3 = 0
5. Gudang 3 ke kota tujuan C ------> R3 + C3 = 19

Persamaan-persamaan tersebut di atas terlihat terdapat 6 variabel yang berbeda, yaitu R1, R2, R3 dan C1, C2, C3. Sehingga untuk kelima persamaan dengan enam variabel yang berbeda, maka untuk menentukan nilai Rid an Cj, maka bisa dengan mengasumsikan salah satu variabel bernilai tertentu untuk digunakan menyelesaikan persamaan lainnya. Yang paling mudah adalah dengan membuat asumsi bernilai nol (0) untuk salah satu variabel. Misalnya dengan mengambil variabel R1 sama dengan nol (R1 = 0), sehingga kita bisa mendapatkan ke-5 sistem persamaan di atas menggunakan substitusi menjadi sebagai berikut:

1. Gudang 1 ke kota tujuan A ------> R1 + C1 = 20
   0 + C1 = 20
   C1 = 20

2. Gudang 1 ke kota tujuan A ------> R2 + C1 = 5
   R2 + 20 = 5
   R2 = -15
3. Gudang 1 ke kota tujuan A ------> R2 + C2 = 20
   -15 + C2 = 20
   C2 = 35
4. Gudang 1 ke kota tujuan A ------> R2 + C3 = 0
   -15 + C3 = 0
   C3= 15
5. Gudang 1 ke kota tujuan A ------> R3 + C3 = 19
   R3 + 15 = 19
   R3 = 4

Langkah Kedua

Pada tahap kedua adalah dengan menempatkan hasil perhitungan setiap variabel ke dalam tabel awal transportasi seperti terlihat dalam tabel berikut ini:

Tabel 6. 17 Penentuan Nilai Baris dan Kolom

		$R_1 = 0$	R2 = -15 Kota tujuan	$R_3 = 4$	
		A	B	C	Persediaan
$C_1 = 20$ Gudang	1	20 / 50	5 / 40	8	90
$C1_2 = 35$	2	15	20 / 60	10	60
$C_3 = 15$	3	25	10 / 10	19 / 40	50
Permintaan		50	110	40	200

Terlihat dalam tabel di atas bahwa masing-masing baris dan kolom telah mempunyai nilai yang akan dipakai untuk menghtung besarnya perubahan per unit biaya total yang dihasilkan dari mengalokasikan unit ke dalam sel non basis pada tahap berikutnya.

Langkah Ketiga

Pada tahap ketiga akan dilakukan proses perubahan pengalokasian unit ke dalam sel-sel kosong pada baris ke-i dan kolom ke-j. Sehingga pada akhirnya nanti bisa dilihat besarnya perubahan per unit biaya total yang bisa dihasilkan sebagai akibat dari perubahan alokasi tadi. Jika nilai perubahan per unit biaya total disimbolkan sebagai P_{ij}, maka perubahan per unit total biaya atau pij bisa didapatkan dengan cara mengurangkan biaya transportasi (B_{ij}) pada sel non basis yang terletak di baris ke i dan kolom ke j dengan nilai baris ke-i dan nilai kolom ke-j yang saling berkorespondensi. Secara mudah, bisa diformulasikan dengan rumus:

$$P_{ij} = B_{ij} - R_i - C_j$$

Dimana:

P_{ij} = besarnya perubahan per unit biaya total pada sel kosong baris ke-i dan kolom ke-j

B_{ij} = biaya transportasi pada sel kosong baris ke-i dan kolom ke-j

R_i = nilai baris ke-i

C_j = nilai kolom ke-j

Misalnya untuk menghitung nilai sel non basis pada gudang 1 dan kota tujuan C atau U_{1C} bisa dilakukan dengan cara:

$$U_{1C} = B_{13} - R_3 - C_1$$
$$U_{1C} = 8 - 4 - 20$$
$$U_{1C} = -16$$

Itu berarti bahwa pengalokasian sebanyak satu unit sel non basis pada gudang 1 ke kota tujuan 3 akan menghasilkan perubahan (pengurangan) total biaya sebesar $16. Dengan cara yang sama kita bisa menentukan besarnya nilai sel sel non basis lainnya:

Sel non basis U_{2C} $= 15 - 0 - 25$
$= -10$

Sel non basis U_{2A} $= 10 - 4 - 35$
$= -29$

Sel non basis U_{3A} $= 25 - 0 - 15$
$= 10$

Nilai-nilai sel non basis di atas, kemudian dimasukkan dalam tabel transportasi seperti tabel berikut:

Tabel 6. 18 Penentuan Nilai Sel Non Basis

		$R_1 = 0$	R2 = -15 Kota tujuan	$R_3 = 4$	
		A	B	C	Persediaan
$C_1 = 20$ Gudang	1	20 / 50	5 / 40	8 / -16	90
$C_2 = 35$	2	-10 / 15	20 / 60	-29 / 10	60
$C_3 = 15$	3	10 / 25	10 / 10	19 / 40	50
Permintaan		50	110	40	200

Sel non basis (U_{ij}) yang berisi nilai negatif mengandung arti bahwa alokasi yang akan bila dilakukan di sel tersebut akan mengurangi biaya pengiriman. Sementara itu, sebaliknya untuk nilai non basis (U_{ij}) yang bertanda positif mengadung arti bahwa alokasi pengiriman di sel tersebut akan menambah ongkos kirim. Sehingga pengisian alokasi harus dilakukan pada sel non basis yang bernilai negatif. Apabila ditemukan terdapat lebih dari satu sel non basis yang bernilai negatif, maka dipilih nilai sel non basis yang bernilai negative terbesar karena berpotensi akan mengurangi biaya kirim paling besar. Berdasarkan tabel hasil langkah ketiga di atas, terdapat tiga sel non basis yang bernilai negatif dan sel U_{2C} (-29) memiliki nilai negatif terbesar di antara sel non negatif lain sehingga sel inilah yang akan menjadi target perbaikan alokasi pada tahap selanjutnya.

Tahap Keempat

Pada tahap ini akan dilakukan perbaikan alokasi pengiriman berdasarkan nilai sel non basis terpilih pada tahap sebelumnya yakni sel U_{2C} (-29). Prosedurnya adalah dengan memilih satu sel basis yang sebaris dengan dengan sel U_{2C} yaitu sel U_{2B} serta memilih satu sel basis yang berada dikolom yang sama dengan U_{2C} yaitu sel U_{3C}. Kedua sel terpilih tersebut selanjutnya akan diberi nilai (tanda) negative (-) sebagai calon sel yang akan dikurang alokasinya untuk dialihkan (ditambahkan) ke sel lain (termasuk sel non basis U_{2C}). Langkah berikutnya adalah memilih satu sel yang sebaris atau sekolom

dengan kedua sel bertanda negative tersebut, yaitu sel U_{3B}. Sel tersebut akan kita beri tanda positif (+) dengan maksud agar saat re-alokasi pengisian nanti sel tersebut akan mendapatkan (bertambah) alokasi yang diambilkan dari dua sel bertanda negative di atas. Sehingga arah pengisian alokasi sel U_{2C} telah mendapatkan prosedur yang jelas seperti tabel di bawah ini :

Tabel 6. 19 Penentuan Nilai Sel Non Basis Terpilih

		$R_1 = 0$	R2 = -15 Kota tujuan	$R_3 = 4$	
		A	B	C	Persediaan
$C_1 = 20$ Gudang	1	20 / 50	5 / 40	8	90
$C_2 = 35$	2	15	20 - / 60	10 +	60
$C_3 = 15$	3	25	10 + / 10	19 - / 40	50
Permintaan		50	110	40	200

Tahap Kelima

Pada tahap ini, dilakukan pengisian kembali atau re-alokasi sel yang bertanda positif yang diambilkan dari sel-sel yang bertanda negatif dan mempunyai alokasi paling sedikit. Dua sel negative adalah sel U_{2B} yang beralokasi 60 unit dan sel U_{3C} yang beralokasi 40 unit, sehingga jumlah alokasi sel U_{3C} yang paling kecil yang akan dipilih menjadi acuan alokasi ke sel-sel sesuai tanda masing-masing sel. Misalnya, untuk sel terpilh U_{2C}(bertanda +) dialokasikan sebanyak 40 unit sedangkan sel U_{3C} (bertanda -) jumlah alokasinya dikurangi sebanyak 40 unit menjadi 0 unit. Lalu sel U_{3B} alokasinya ditambah sebanyak 40 unit sehingga menjadi 50 unit dan akhirnya sel U_{2B} (bertanda -) akan dikurangi 40 unit sehingga menjadi 20 unit. Detail alokasi baru hasil perbaikan tahap ini bisa dilihat sebagai berikut:

Tabel 6. 20 Re-Alokasi Berdasarkan Nilai Sel Non Basis Terpilih

		$R_1 = 0$	R2 = -15 Kota tujuan	$R_3 = 4$	
		A	B	C	Persediaan
$C_1 = 20$ Gudang	1	20 / 50	5 / 40	8	90
$C_2 = 35$	2	15	20 / 20	40 / 10	60
$C_3 = 15$	3	25	50 / 10	0 / 19	50
Permintaan		50	110	40	200

Langkah di atas diulangi hingga tidak ditemukan lagi nilai negative sel non basis Uij. Dengan kata lain jika semua sel non basis Uij sudah bernilai 0 atau bertanda positif maka penyelesaian optimal sudah tercapai dan pengisian kembali tidak diperlukan lagi. Untuk membuktikan bahwa re-alokasi yang sudah dialukan bisa mengurangi biaya pengiriman total bisa dilakukan penghitungan total biaya kirim sebagai berikut:

Biaya kirim dari gudang 1 ke tujuan A = 50 unit x $ 20/unit
= $ 1.000

Biaya kirim dari gudang 1 ke tujuan B = 40 unit x $ 5/unit
= $ 200

Biaya kirim dari gudang 2 ke tujuan B = 20 unit x $ 20/unit
= $ 400

Biaya kirim dari gudang 2 ke tujuan C = 40 unit x $ 10/unit
= $ 400

Biaya kirim dari gudang 3 ke tujuan B = 50 unit x $ 0/unit
= $ 0

Biaya kirim dari gudang 3 ke tujuan C = 10 unit x $ 19/unit
= $ 190

Jadi total biaya kirim yang harus dikeluarkan setelah dilakukan perbaikan alokasi pengiriman dengan menggunakan MODI sejauh ini sebesar $ 2.190. Total biaya kirim hasil modifikasi alokasi dengan MODI pada tahap ini telah terbukti mampu menurunkan total biaya kirim dari kondisi awal sebelum dilakukan modifikasi. Dari sebelum dilakukan

modifikasi dengan MODI sebesar $3.800 menjadi $2.190 atau terjadi penurunan biaya kirim sebesar $ 1.610 atau sebesar 40%.

Penyimpangan Dalam Problem Transportasi

Dalam penyelesaian problem transportasi, tidak semua masalah kita jumpai dalam bentuk standard atau umum yang bisa kita selesaikan dengan cara standard seperti yang telah dijelaskan dalam bab ini sebelumnya. Ada kalanya kita jumpai penyimpangan atau *"unusual"* problem transportasi sehingga diperlukan beberapa tambahan prosedur yang lebih kompleks dalam penyelesaiannya. Beberapa penyimpangan yang sering terjadi dalam problem transportasi antara lain:

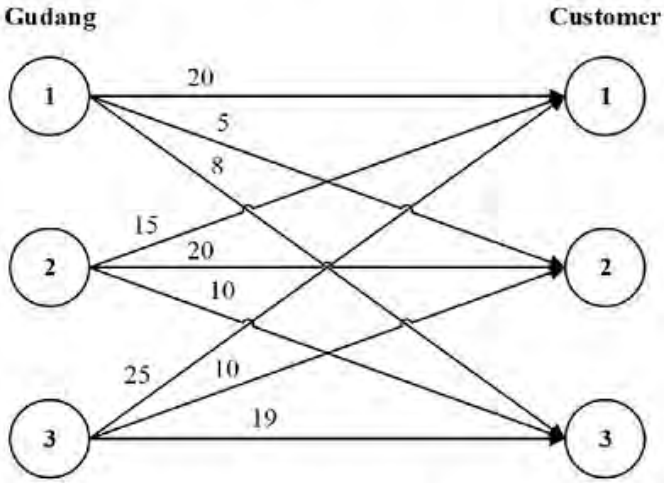
- Terdapat kondisi dimana jumlah kapasitas sumber */source* yang tidak sama dengan jumlah permintaan atau demand tujuan. Sehingga jumlah kapasistas sumber dan jumlah total permintaan kota tujuan tidak seimbang. Ketidakseimbangan ini bisa terjadi pada sisi jumlah kapasitas *supply* dari sumber yang lebih besar daripada jumlah total permintaan kota tujuan atau ketidakseimbangan yang terjadi sebaliknya. Jika terjadi kasus demikian, maka kita bisa melakukan penyeimbangan kapasitas sumber dan jumlah permintaan agar sama jumlahnya sebelum kita selesaikan dengan metode transportasi. Proses penyeimbangan ini sering kita sebut sebagai proses penambahan kolom *dummy/slack* jika kapasitas sumber *supply* lebih besar dari jumlah permintaan dan penambahan baris *dummy/slack* jika jumlah permintaan semua kota tujuan lebih besar dari kapasistas total semua sumber *supply*. Besaran nilai *dummy* dari sumber *supply* dan dummy permintaan kota tujuan disesuaikan sedemikian rupa sehingga terjadi keseimbangan antara jumlah total sumber *supply* dan jumlah permintaan semua kota tujuan. Sedangkan untuk biaya kirm di tiap sel-nya untuk baris atau kolom *dummy* ini bernilai nol (0) karena memang kolom/baris *dummy* ini tidak rill. Kalau sudah dilakukan penambahan *dummy* sesuai kebutuhan dan sudah terjadi keseimbangan, maka tabel awal transportasi bisa dibentuk kembali dan siap dilakukan penyelesaian seperti biasanya.
- Dalam kenyataan riil-nya, masalah penyelesaian problem transportasi ini tidak selalu *smooth* sesuai dengan yang kita rencanakan terutama menyangkut akses transportasi. Sering dijumpai kasus dimana terjadi

ketidaklancaran proses distribusi karena adanya kemacetan atau pemblokiran atau pembatasan akses transportasi dari satu sumber ke daerah tujuan yang diinginkan. Sehingga meskipun pada kondisi tertentu daerah kota tujuan tersebut memiliki biaya kirim yang paling rendah (murah) namun tidak bisa dilakukan alokasi. Pada kasus ini biasanya sel-sel yang tidak bisa dilakukan alokasi tidak dipilih untuk diberi alokasi walaupun biayanya paling murah dan bisa langsung dilanjutkan ke proses pemilihan biaya termurah berikutnya untuk dilakukan alokasi.

- Seringkali dalam proses alokasi dari sumber *supply* ke daerah tujuan terdapat kebijakan prioritas. Kebijakan prioritas ini merupakan kebijakan mendahulukan alokasi untuk daerah tertentu. Kebalikan dari kasus pemblokiran di atas, problem prioritas ini memperbolehkan melakukan alokasi pertama kali ke daerah tujuan yang diinginkan meskipun biaya kirimnya lebih mahal. Hal ini sering kita jumpai saat metode transportasi ini digunakan untuk men-*supply* daerah tujuan yang sedang mengalami bencana atau biasa dikenal sebagai *humanitarian transportation operations* atau transportasi untuk pemulihan bencana.
- Pada umumnya, problem transportasi yang kita jumpai adalah problem minimasi biaya kirim total dari sumber *supply* ke kota tujuan. Namun tidak tertutup kemungkinan kita mendapati fungsi tujuan maksimasi. Dalam perkembangannya, problem transportasi telah berkembang ke arah yang lebih kompleks termasuk memecahkan permasalahan transportasi untuk kasus-kasus fungsi tujuan maksimasi. Beberapa artikel terkait problem maksimasi fungsi tujuan dalam metode transportasi misalnya memaksimalkan laba (Sakawa, Nishizaki, & Uemura, 2001; Spasovic, Boile, & Bladikas, 1994), penentuan jadwal produksi agar hasil produksi maksimum (Kantorivich, 1939) dan lain sebagainya.

Penyelesaian Masalah Transportasi Dengan R

Sebagai contoh, saat kita akan mengirim barang ke 3 kostumer berbeda dengan permintaan yang berbeda. Barang tersebut disimpan pada 3 gudang dengan kapasitas berbeda juga. Pertanyaan yang perlu dijawab adalah dari mana barang harus dikirim untuk menghasilkan biaya paling minimum.



Gambar 6. 3 Contoh Problem Transportasi

Model matematis :

Model matematis :

$$Min\ Z = [20X_{11} + 5X_{12} + 8X_{13}] + [15X_{21} + 20X_{22} + 10X_{23}] + [25X_{31} + 10X_{22} + 19X_{33}]$$

S.T.

$$X_{11} + X_{12} + X_{13} \leq 90$$
$$X_{21} + X_{22} + X_{23} \leq 60$$
$$X_{31} + X_{32} + X_{33} \leq 50$$
$$X_{11} + X_{21} + X_{31} \geq 50$$
$$X_{12} + X_{22} + X_{32} \geq 110$$
$$X_{13} + X_{23} + X_{33} \geq 40$$
$$\sum X_{ij} \geq 0$$

Model di atas akan diselesaikan dengan bahasa R dengan *syntax* sebagai berikut :

```
#EXAMPLE Transportation
> f.obj<-c(20,5,8,15,20,10,25,10,19)
> f.con<-matrix(c(1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0
,1,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,1),nrow=6,byrow=TRUE)
> f.dir<-c("<=","<=","<=",">=",">=",">=")
> f.rhs<-c(90,60,50,50,110,40)
> #
```

```
> #Solution
> #
> lp("min",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)
Error in lp("min", f.obj, f.con, f.dir, f.rhs, all.int = TRUE) :
  could not find function "lp"
> library(lpSolve)
> f.obj<-c(20,5,8,15,20,10,25,10,19)
> f.con<-matrix(c(1,1,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0
,1,0,0,1,0,0,0,1,0,0,1,0,0,1,0,0,0,1,0,0,1,0,0,1),nrow=6,byrow=TRUE)
> f.dir<-c("<=","<=","<=",">=",">=",">=")
> f.rhs<-c(90,60,50,50,110,40)
> #
> #Run Solution
> #
> lp("min",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)
Success: the objective function is 1890
> lp("min",f.obj,f.con,f.dir,f.rhs,all.int=TRUE)$solution
[1]  0 60 30 50  0 10  0 50  0
```

BAB 7
MODEL TRANSHIPMENT

Pengantar Pemrograman Linier : Model Transhipment
Model Dasar Metode Transhipment
Penyelesaian Program Linier Metode Transhipment
Penyelesaian Masalah Transhipment Dengan R

MODEL TRANSHIPMENT

Pengantar Pemrograman Linier : Model Transhipment

Dalam model transportasi yang telah dibahas sebelumnya, terlihat bahwa rute pengiriman dilakukan langsung dari sumber supply yang bisa berupa pabrik, gudang atau warehouse ke daerah tujuan yang bisa berupa pusat distribusi (*distribution centre*), retailer atau konsumen. Proses routing metode transportasi dari sumber supply ke daerah tujuan punya tujuan untuk meminimalkan biaya pengirman, sehingga pengiriman langsung dalam metode transprtasi ini sangat identik dengan pendekatan jarak terpendek alias biaya terkecil. Apabila dalam kasus problem transportasi yang kita hadapi terdapat hanya bebarapa sumber supply dan beberapa daerah tujuan saja maka penyelesaiannya tidak terlalu rumit. Namun bila ternyata jumlah sumber supply dan jumlah daerah tujuan yang sedang dihadapi dalam jumlah yang cukup besar, maka penyelesaiannya akan lebih rumit jika digunakan metode transportasi biasa. Oleh karena itu diperlukan sebuah pendekatan lebih khusus dan sistematik untk menyelesaikannya. Salah satu pendekatan yang bisa dipakai adalah dengan membentuk problem "besar" tersebut menjadi model transit atau *tanshipment model* yang merupakan pengembangan dari kasus khusus transportasi.

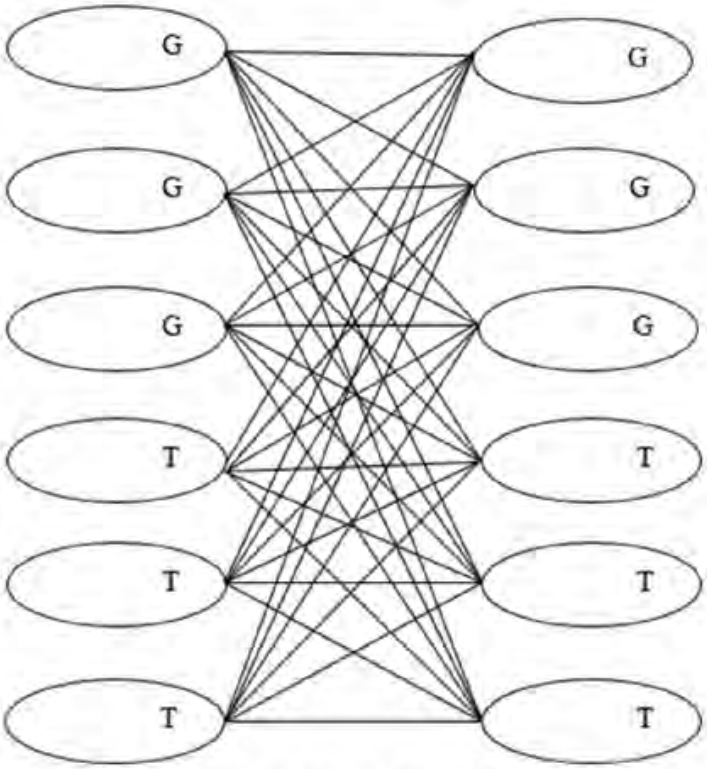
Model Transhipment memungkinkan menggunakan titik atau node perantara dalam mensupply barang sebagai tujuan perantara

sebelum pengiriman sampai ke daerah tujuan yang sebenarnya. Sebagai contoh, dalam proses pengiriman barang dari sebuah sumber (pabrik) ke suatu daerah tujuan tidak dilakukan secara langsung ke tujuan tersebut namun bisa dikirim ke titik transit atau gudang penyimpanan di tempat lain lebih dulu. Model transit ini akan secara otomatis menentukan rute antara sebuah sumber supply dan suatu daerah tujuan dengan biaya minimum tanpa harus memenetukan rutenya terlebih dahulu. Dengan pendekatan transhipment ini kita dapat menentukan jumlah yang akan dikirm dari satu lokasi ke lokasi lain sehingga semua permintaan atau demand dapat dipenuhi dengan biaya minimum.

Pada model transhipment ini, pada dasarnya semua jumlah supply dari semua sumber bisa dialirkan ke sembarang sumber atau tujuan sebelum didistribusikan kembali ke tempat lain. Hal tersebut menunjukkan bahwa setiap titik atau node dalam jaringan metode transportasi baik itu sumber supply maupun daerah tujuan dapat dianggap sebagai sumber langsung atau tujuan langsung. Apabila tidak diketahui mana titik yang menjadi sumber atau titik mana yang merupakan tujuan, maka kita dapat membentuk model jaringan yang menganggap bahwa setiap titik dapat diperspektifkan sebagai sumber dan tujuan sekaligus. Atau dengan kata lain, dalam metode transhipment, jumlah sumber atau tujuan akan sama dengan jumlah sumber atau tujuan model transportasi (Agustini & Rahmadi, 2004).

Model Dasar Metode Transhipment

Untuk membentuk model dasar metode transhipment, kita gunakan contoh soal sebelumnya. Dalam contoh soal sebelumnya, diketahui bahwa terdapat tiga gudang dan tiga daerah tujuan dengan kapasitas supply dan permintaan yang seimbang. Jaringan antara ketiga sumber supply dan ketiga daerah tujuan tersebut dapat ditunjukkan pada gambar berikut ini :



Gambar 7. 1 Jaringan Model Transhipment

Dari model jaringan transhipment di atas, terlihat bahwa baik sumber supply maupun daerah tujuan dimungkinkan melakukan aktivitas pengiriman langsung dari setiap titik. Sehingga persediaan tambahan diperlukan di setiap sumber supply dan daerah tujuan. Persediaan tambahan atau lebih dikenal sebagai *buffer stock* (BS) ini berfungsi sebagai persediaan pendukung yang jumlahnya paling sedikit sama dengan jumlah permintaan dari daerah tujuan yang ada di model transportasi standard. Bentuk matematis yang bisa dibuat untuk menggambarkan kebutuhan *buffer stock* (BS) tergantung dari keseimbangan antara kapasitas suber supply dan jumlah permintaan. Jika terjadi pola ketidakseimbangan dimana jumlah demand (Σai) tidak sama dengan jumlah supply (Σbj), maka *buffer stock* pada kejadian ini dirumuskan sebagai:

$$BS = \text{maks}\{\Sigma ai, \Sigma bj\}$$

Dimana BS = *buffer stock*,

ai = supply sumber i,

bj = permintan tujuan j.

Sementara itu, bentuk matematis dari *buffer stock* bila jumlah antara kapasitas supply sumber dan jumlah demand semua daerah tujuan sama atau terjadi keseimbangan, maka besarnya persediaan pendukung atau buffer stock (BS) akan bernilai paling sedikit sama dengan jumlah supply sumber (atau permintaan) dari model transportasi standard. Secara matematis, bisa digambarkan dalam rumus berikut ini:

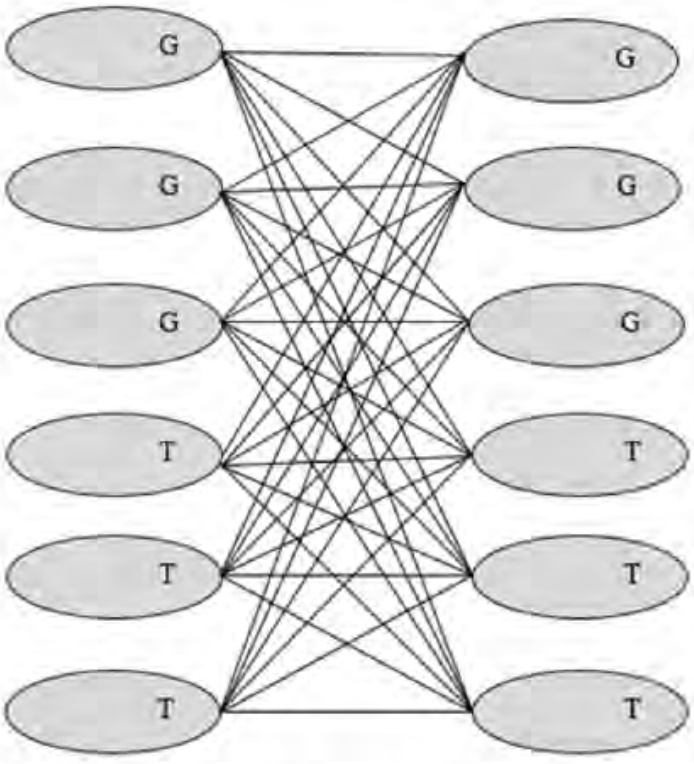
$$BS \geq \Sigma ai = \Sigma bj$$

Dimana BS = buffer stock,

ai = supply sumber i,

bj = permintan tujuan j.

Oleh karena itulah untuk problem transhipment dengan kondisi seimbang gambar jaringan transhipment gambar di atas (sesuai dengan soal metode transportasi sebelumnya) bisa digambarkan besaran nilai masing-masing titik sumber supply dan permintaan dari setiap titik daerah tujuan sebagai berikut :



Gambar 7. 2 Buffer stock untuk model transhipment

Untuk data biaya kirim, biaya kirim dari dari sumber asli ke daerah tujuan sebenarnya tetap seperti data semula, sementara unit biaya lain seperti biaya kirim dari sumber supply ke sumber supply itu sendiri (dirinya sendiri) atau biaya kirim dari daerah tujuan ke daerah tujuan itu sendiri (dirinya sendiri) adalah sebesar 0 (nol). Dalam prinsip model transhipment, rute distribusi menjadi pertimbangan utama saat memutuskan menggunakan pendekatan ini, sehingga untuk biaya kirim dari satu sumber ke daerah tujuan dan arah sebaliknya mestinya tidak sama. Misalnya biaya kirim dari gudang 1 ke daerah tujuan 1 tidak sama dengan biaya kirim dari daerah tujuan 1 ke gudang 1. Hal ini lebih disebabkan karena biaya kirim sangat identik dengan jarak dan rute, sehingga rute dari satu titik ke titik lainnya dianggap berbeda atau tidak sama dengan rute sebaliknya. Begitu juga untuk data biaya kirim dari satu daerah tujuan ke daerah tujuan lain dianggap berbeda dengan biaya kirim arah sebaliknya karena rute yang berbeda. Sehingga dengan asumsi nilai buffer stock (BS) = sebesar 40 unit dan biaya kirim diasumsikan seperti terlihat di tabel maka tabel model transhipment akan berbentuk sebagai berikut:

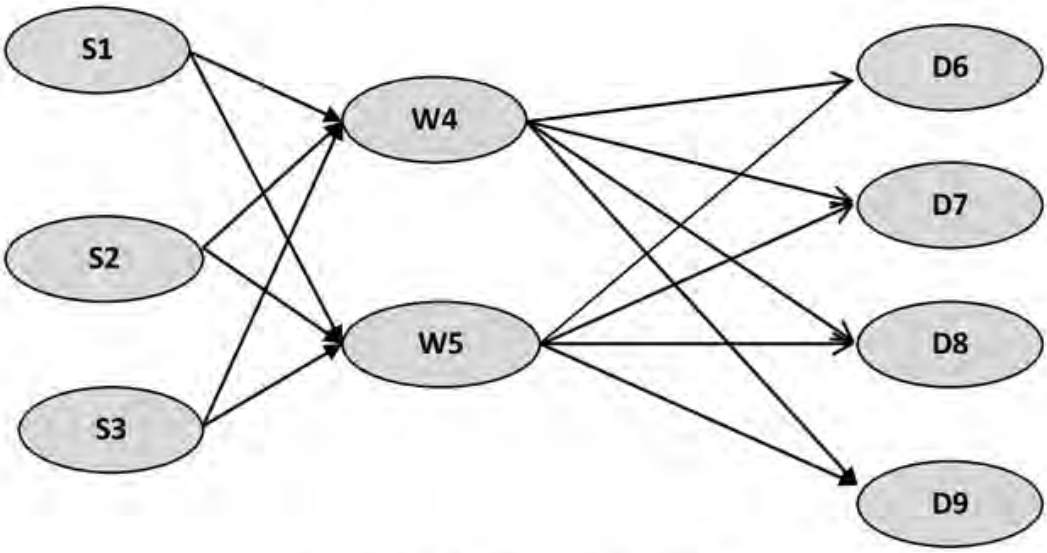
Tabel 7. 1 Model Transit

Tujuan / Supply	Gudang 1	Gudang 2	Gudang 3	Tujuan A	Tujuan B	Tujuan C
Gudang 1	$0 40 unit	$5	$8	$20	$5	$8
Gudang 2	$7	$0 40 unit	$6	$15	$20	$10
Gudang 3	$9	$7	$0 40 unit	$25	$10	$19
Tujuan A	$20	$12	$20	$0 40 unit	$7	$8
Tujuan B	$3	$15	$12	$6	$0 40 unit	$9
Tujuan C	$7	$12	$20	$10	$8	$0 40 unit

Dari tabel di atas, terlihat bahwa sel-sel searah diagonal merupakan hasil dari pemakaian persediaan pendukung atau *buffer stock*. Hasil yang terlihat di tabel tersebut bukanlah merupakan informasi hasil final atau optimal. Dalam tabel tersebut terlihat bahwa alokasi pengiriman antara gudang 1 ke gudang 1 sendiri sebanyak BS-nya atau 40 unit. Artinya gudang 1 dikirim alokasi sebanyak 40 unit dari gudang 1, sementara sisanya yang lain dikirim langsung dari gudang penyimpanan ke daerah tujuan. Perlu diingat bahwa persediaan pendukung atau *buffer stock* ditambahkan hanya bila suatu titik bertindak sebagai sumber sekaligus daerah tujuan.

Penyelesaian Program Linier Metode Transhipment

Pada model transhipment, ada beberapa kasus dimana terdapat titik perantra sebelum pengiriman dilakukan ke daerah tujuan akhir. *Buffer stock* akan disimpan lebih dulu di tempat penyimpanan sementara di gudang atau di pabrik sebelum dikirim ke retailer, pedagang eceran atau agen. Sebagai contoh, misalnya terdapat tiga sumber supply berupa pabrik yang berlokasi di tiga tempat berbeda (S1, S2 dan S3) yang akan didistribusikan ke empat retailer yang berbeda (D6, D7, D8 dan D9). Dalam proses pengiriman tersebut direncanakan produk yang akan di kirim ke tujuan akhir akan disimpan di dua *warehouse* atau gudang 106 yimpanan di kota yang berbeda (W4 dan W5). Jaringan distribusinya bisa dilihat di dalam gambar di bawah ini:



Gambar 7. 3 Jaringan Distribusi

Adapun unit biaya transportasi dari satu lokasi ke lokasi yang lain disajikan dalam tabel berikut ini:

Tabel 7. 2 Unit Biaya Transportasi Dari Pabrik Ke Warehouse

Pabrik	Biaya kirim dari pabrik ke gudang penyimpanan ($)/unit	
	Warehouse 4	Warehouse 5
Pabrik 1 (S1)	6	8
Pabrik 2 (S2)	8	12
Pabrik 3 (S3)	10	5

Sementara itu, data biaya transportasi per unit dari gudang penyimpanan ke lokasi tujuan akhir (retailer) bisa dilihat di dalam tabel di bawah ini:

Tabel 7. 3 Unit Biaya Transportasi Dari Warehouse Ke Retailer

Pabrik	Biaya kirim gudang penyimpanan ke retailer ($)/unit			
	Retailer 6	Retailer 7	Retailer 8	Retailer 9
Warehouse 1 (W4)	9	7	6	10
Warehouse 2 (W5)	7	9	6	8

Sedangkan data permintaan dan kapasitas sumber supply dari masing-masing retailer dan pabrik adalah sebagai berikut:

Tabel 7. 4 Kapasitas Pabrik

Pabrik	Kapasitas (unit)
Pabrik 1	400
Pabrik 2	450
Pabrik 3	350

Sementara jumlah permintaan dari masing-masing kota tujuan (retailer) bisa dilihat dalam tabel berikut ini:

Tabel 7. 5 Jumlah Permintaaan Warehouse

Warehouse	Permintaan (unit)
Retailer 6	200
Retailer 7	500
Retailer 8	300
Retailer 9	200

Berdasarkan data-data di atas, selanjutnya kita bisa menyusun sebuah sistem persamaan program linier. Dengan menggunakan prinsip langkah-langkah dalam program linier maka kita bisa mendefinisikan variabel keputusan sebagai variable Xij yang menunjukkan jumlah unit produk yang akan di kirim dari titik I ke titik j. Misalnya X_{14} menunjukkan jumlah unit produk yang akan dikirim ke dari pabrik 1 ke warehouse 4, X_{46} menunjukkan jumlah unit produk yang akan dikirimkan dari warehouse 2 ke retailer 6 dan seterusnya. Bentuk lengkap persamaan matematisnya bisa dibentuk seperti berikut ini:

Persamaan matematis untuk kapasitas pabrik 1 (400 unit):

$$X_{14} + X_{15} \leq 400$$

Persamaan ini menerangkan bahwa jumlah unit yang akan dikirim dari pabrik 1 ke masing-masing warehouse 4 dan 5 tidak boleh melebihi 400 unit.

Persamaan matematis untuk kapasitas pabrik 2 (450 unit):

$$X_{24} + X_{25} \leq 450$$

Persamaan ini mengkondisikan formulasi tersebut agar jumlah unit yang akan dikirim dari pabrik 2 ke masing-masing warehouse 4 dan 5 tidak boleh melebihi 450 unit.

Persamaan matematis untuk kapasitas pabrik 2 (350 unit):

$$X_{34} + X_{35} \leq 350$$

Equation di atas menerangkan bahwa jumlah unit yang akan dikirim dari pabrik 3 ke masing-masing warehouse 4 dan 5 tidak boleh melebihi 350 unit.

Pertimbangan terkait keseimbangan supply dan demand tersebut juga akan dilakukan dalam membentuk persamaan yang menghubungkan warehouse dan retailer. Jumlah yang unit yang dikirimkan dari masing-masig warehouse 3 dan 4 ke retailer yang ada harus sama dengan jumlah unit yang dikirimkan ke kedua warehouse tersebut. Atau dengan kata lain:

Jumlah (unit) yang dikirm ke warehouse 4 = jumlah unit yang disalurkan dari warehouse 4. Aatau dengan persamaan matematis liniernya kita bisa membentuknya menjadi:

$$X_{14} + X_{24} + X_{34} = X_{46} + X_{47} + X_{48} + X_{49}$$

Atau persamaan liniernya menjadi:

$$-X_{14} - X_{24} - X_{34} + X_{46} + X_{47} + X_{48} + X_{49} = 0$$

Dengan cara yang sama kita bisa menentukan persamaan liniernya untuk warehouse 5 menjadi:

$$X_{15} + X_{25} + X_{35} = X_{56} + X_{57} + X_{58} + X_{59}$$

Atau bentuk persamaan liniernya menjadi:

$$-X_{15} - X_{25} - X_{35} + X_{56} + X_{57} + X_{58} + X_{59} = 0$$

Sementara itu, kendala atau batasan yang berkaitan dengan kota tujuan atau retailer adalah permintaan yang diminta oleh masing-masing retailer. Misalnya, retailer 6 meminta jumlah barang sejumlah 200 unit, maka jumlah yang harus dikirim dari kedua warehouse penyimpanan ke retailer harus sama dengan 200 unit. Atau bentuk persamaan liniernya akan menjadi:

$$X_{46} + X_{56} = 200$$

Persamaan tersebut menunjukkan bahwa jumlah unit yang dikirm dari warehouse 4 dan warehouse 5 ke retailer 6 harus sama dengan 200 unit. Dengan cara yang sama, kita bisa menentukan persamaan linier untuk retailer 7, 8 dan 9 sebagai berikut:

Retailer 7: $X_{47} + X_{57} = 500$

Retailer 8: $X_{48} + X_{58} = 300$

Retailer 9: $X_{49} + X_{59} = 200$

Sedangkan untuk persamaan fungsi tujuan dalam kasus transhipment ini adalah untuk meminmalkan total biaya transportasi dari seluruh pengiriman. Dengan mengalikannya dengan biaya kirim per unit dari satu titik ke titik yang lain maka fungsi tujuan bisa dibentuk menjadi sebuah persamaan sebagai berikut:

Fungsi tujuan: $6X_{14} + 8X_{15} + 8X_{24} + 12X_{25} + 10X_{34} + 5X_{35} + 9X_{46} + 7X_{47} + 6X_{48} + 10X_{49} + 7X_{56} + 9X_{57} + 6X_{58} + 8X_{59}$

Sehingga penggabungan keseluruhan persaamaan kendala dan fungsi tujuan dalam problem transhipment ini menjadi sebuah sistem persamaan programa linier dengan 9 fungsi kendala adalah sebagai berikut;

Minimalkan $6X_{14} + 8X_{15} + 8X_{24} + 12X_{25} + 10X_{34} + 5X_{35} + 9X_{46} + 7X_{47} + 6X_{48} + 10X_{49} + 7X_{56} + 9X_{57} + 6X_{58} + 8X_{59}$

Dengan kendala:

1). $X_{14} + X_{15} \leq 400$
2). $X_{24} + X_{25} \leq 450$
3). $X_{34} + X_{35} \leq 350$
4). $-X_{14} - X_{24} - X_{34} + X_{46} + X_{47} + X_{48} + X_{49} = 0$
5). $-X_{15} - X_{25} - X_{35} + X_{56} + X_{57} + X_{58} + X_{59} = 0$
6). $X_{46} + X_{56} = 200$
7). $X_{47} + X_{57} = 500$
8). $X_{48} + X_{58} = 300$
9). $X_{49} + X_{59} = 200$

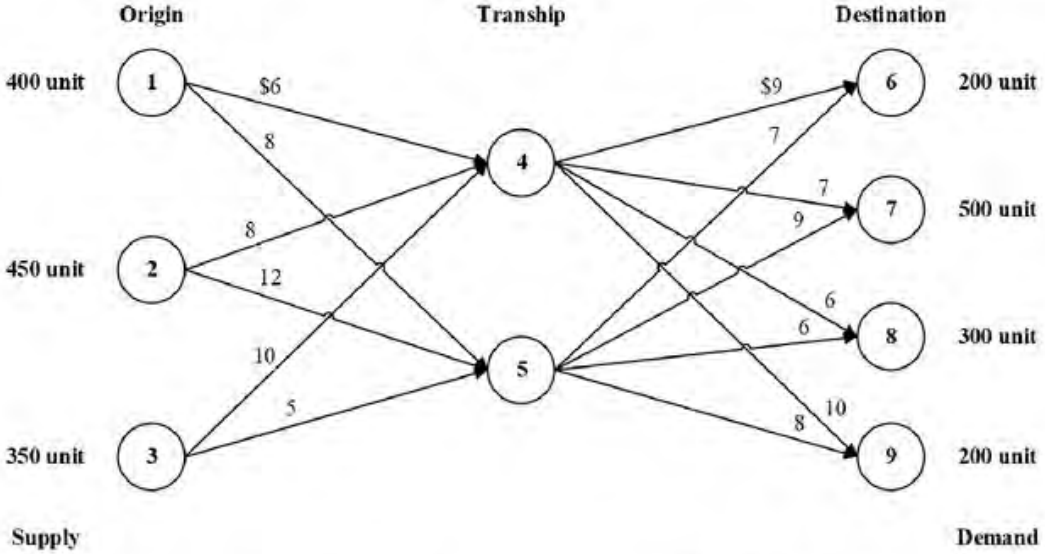
$Xij \geq 0$ untuk semua i dan j.

Hasil perhitungan atau solusi dari sistem persamaan program linier problem transhipment di atas menghasilkan solusi optimal sebagai berikut:

$X_{14} =$	$X_{47} =$
$X_{15} =$	$X_{48} =$
$X_{24} =$	$X_{49} =$
$X_{25} =$	$X_{56} =$
$X_{34} =$	$X_{57} =$
$X_{35} =$	$X_{58} =$
$X_{46} =$	$X_{59} =$

Penyelesaian Masalah Transhipment Dengan R

Sebagai contoh, saat kita akan mengirim barang ke 4 customer berbeda dengan permintaan yang berbeda. Barang tersebut disimpan pada 3 gudang dengan kapasitas berbeda juga. Pengiriman harus melalui 2 gudang penyangga. Bagaimana skenario pengiriman barang yang menghasilkan biaya paling minimum.



Gambar 7. 4 Contoh Problem Jaringan Transhipment

Model matematis :

$$Min\ Z = 6X_{14} + 8X_{15} + 8X_{24} + 12X_{25} + 10X_{34} + 5X_{35}$$
$$+9X_{46} + 7X_{47} + 6X_{48} + 10X_{49}$$
$$+7X_{56} + 9X_{57} + 6X_{58} + 8X_{59}$$

S.T.

$$X_{14} + X_{15} = 400$$
$$X_{24} + X_{25} = 450$$
$$X_{34} + X_{35} = 350$$
$$-X_{14} - X_{24} - X_{34} + X_{46} + X_{47} + X_{48} + X_{49} = 0$$
$$-X_{15} - X_{25} - X_{35} + X_{56} + X_{57} + X_{58} + X_{59} = 0$$
$$X_{46} + X_{56} = 200$$
$$X_{47} + X_{57} = 500$$
$$X_{48} + X_{58} = 300$$
$$X_{49} + X_{59} = 200$$
$$X_{ij} \geq 0$$

Model di atas akan diselesaikan dengan bahasa R dengan *syntax* sebagai berikut:

```
#EXAMPLE Transhipment
#Gunakan Library "lpSolveAPI"
```

```
> library(lpSolveAPI)

#make.lp(jumlah constraints, jumlah variabel keputusan, error reporting)
> lpmodel<-make.lp(10, 14, "full")

#Input Fungsi Tujuan
> set.objfn(lpmodel, c(6,8,8,12,10,5,
+                      9,7,6,10,7,9,6,8))

#Input Constraints
#Supply Constraints
> add.constraint(lpmodel, c(1,1,0,0,0,0,
+                           0,0,0,0,0,0,0,0),"=",400)
> add.constraint(lpmodel, c(0,0,1,1,0,0,
+                           0,0,0,0,0,0,0,0),"=",450)
> add.constraint(lpmodel, c(0,0,0,0,1,1,
+                           0,0,0,0,0,0,0,0),"=",350)
#Transhipment Constraints
> add.constraint(lpmodel, c(-1,0,-1,0,-1,0,
+                           1,1,1,1,0,0,0,0),"=",0)
> add.constraint(lpmodel, c(0,-1,0,-1,0,-1,
+                           0,0,0,0,1,1,1,1),"=",0)
#Demand Constraints
> add.constraint(lpmodel, c(0,0,0,0,0,0,
+                           1,0,0,0,1,0,0,0),"=",200)
> add.constraint(lpmodel, c(0,0,0,0,0,0,
+                           0,1,0,0,0,1,0,0),"=",500)
> add.constraint(lpmodel, c(0,0,0,0,0,0,
+                           0,0,1,0,0,0,1,0),"=",300)
> add.constraint(lpmodel, c(0,0,0,0,0,0,
```

```
+                    (0,0,0,1,0,0,0,1),"=",200)
#Non Negativity Constraints
> add.constraint(lpmodel, c(1,1,1,1,1,1,
+                    1,1,1,1,1,1,1,1),">=",0)

#Setting Variabel Integer
> columns<-seq(1, 14)
> set.type(lpmodel, columns, type = c("integer"))

#Solve
> solve(lpmodel)
Model name:  '' - run #1
Objective:   Minimize(R0)

SUBMITTED
Model size:      20 constraints,     14 variables,          42 non-zeros.
Sets:                          0 GUB,               0 SOS.

 CONSTRAINT CLASSES
General INT      10
General BIN      10

Using DUAL simplex for phase 1 and PRIMAL simplex for phase 2.
The primal and dual simplex pricing strategy set to 'Devex'.

Optimized BLAS was successfully loaded for bfp_LUSOL.
Found dual solution with 1 fixed slack variables left basic.
Optimal solution with dual simplex at iter          8.

Relaxed solution            16150 after         8 iter is B&B base.
```

Primal objective:

Column name	Value	Objective	Min	Max
C1	6	2100	6	8
C2	8	400	6	8
C3	8	3600	-1e+030	10
C4	12	0	10	1e+030
C5	10	0	3	1e+030
C6	5	1750	-1e+030	12
C7	9	0	9	1e+030
C8	7	3500	-1e+030	11
C9	6	1800	-1e+030	8
C10	10	0	10	1e+030
C11	7	1400	-1e+030	7
C12	9	0	5	1e+030
C13	6	0	4	1e+030
C14	8	1600	-1e+030	8

Primal variables:

Column name	Value	Slack	Min	Max
C1	350	0	-1e+030	1e+030
C2	50	0	-1e+030	1e+030
C3	450	0	-1e+030	1e+030
C4	0	2	-350	50
C5	0	7	-50	350
C6	350	0	-1e+030	1e+030
C7	0	0	-350	50
C8	500	0	-1e+030	1e+030

C9	300	0	-1e+030	1e+030
C10	0	0	-350	50
C11	200	0	-1e+030	1e+030
C12	0	4	-50	350
C13	0	2	-50	300
C14	200	0	-1e+030	1e+030

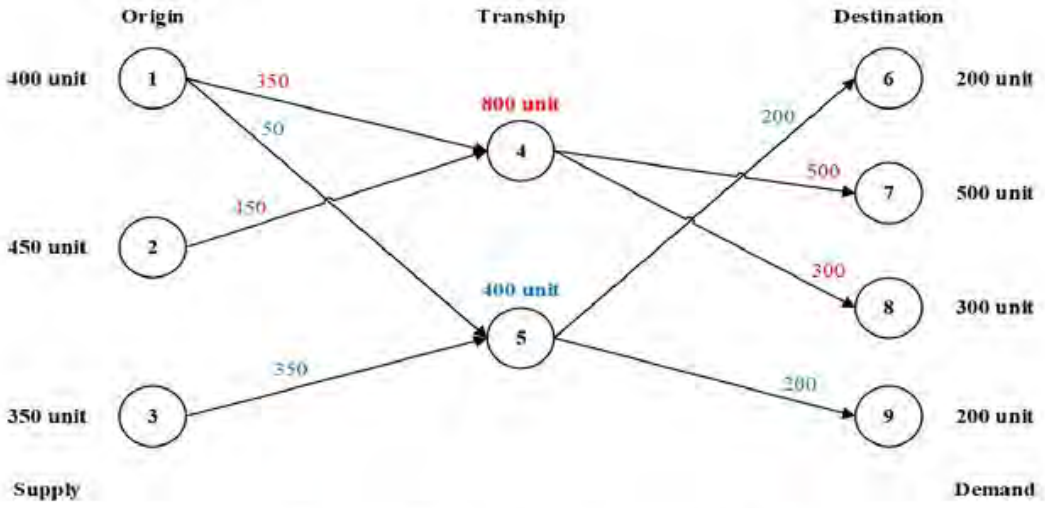
Dual variables:

Row name	Value	Slack	Min	Max
R1	0	0	-1e+030	1e+030
R2	0	0	-1e+030	1e+030
R3	0	0	-1e+030	1e+030
R4	0	0	-1e+030	1e+030
R5	0	0	-1e+030	1e+030
R6	0	0	-1e+030	1e+030
R7	0	0	-1e+030	1e+030
R8	0	0	-1e+030	1e+030
R9	0	0	-1e+030	1e+030
R10	0	0	-1e+030	1e+030
R11	6	400	400	400
R12	8	450	450	450
R13	3	350	350	350
R14	0	0	-1e+030	1e+030
R15	-2	0	0	0
R16	9	200	200	200
R17	7	500	500	500
R18	6	300	300	300
R19	10	200	200	200
R20	0	2400	-1e+030	1e+030

```
#Tampilkan Ringkasan Nilai Fungsi Tujuan
> get.objective(lpmodel)
[1] 16150

#Tampilkan Ringkasan Nilai Masing-Masing Variabel
> get.variables(lpmodel)
 [1] 350  50 450   0   0 350   0 500 300   0 200   0   0 200
```

Dari hasil tersebut dapat disimpulkan biaya minimum yang harus dikeluarkan sebesar $16150 dengan skenario pengiriman seperti pada gambar di bawah ini.



Gambar 7. 5 Rute yang Dihasilkan

DAFTAR PUSTAKA

Agustini, D. H., & Rahmadi, Y. E. (2004). *Riset operasional (konsep-konsep dasar)*. Jakarta: Rineka Cipta.

Handoyo, S., Prasojo, A. P. S., & Naba, A. (2017). *Sistem Fuzzy Terapan dengan Software R*: Universitas Brawijaya Press.

Kantorivich, L. (1939). Mathematical Methods in the Organization and Planning of Production. *Publication House of the Leningrad State University. Translated in Management Science*, 6(1960), 366-422.

Mananoma, T., & Soetopo, W. (2009). Pemodelan Sebagai Sarana Dalam Mencapai Solusi Optimal. *Jurnal Teknik Sipil Universitas Atma Jaya Yogyakarta, 8(3)*, pp. 184-192.

Sakawa, M., Nishizaki, I., & Uemura, Y. (2001). Fuzzy programming and profit and cost allocation for a production and transportation problem. *European Journal of Operational Research, 131(1)*, 1-15.

Simatupang, T. (2000). System Modeling (Pemodelan Sistem). *Nindika, Klaten, Indonesia*.

Simbolon, L. D., Situmorang, M., & Napitupulu, N. (2014). Aplikasi Metode Transportasi Dalam Optimasi Biaya Distribusi Beras Miskin (Raskin) pada Perum Bulog Sub Divre Medan. *saintia Matematika, 2(3)*, 299-311.

Spasovic, L., Boile, M. P., & Bladikas, A. K. (1994). Bus transit service coverage for maximum profit and social welfare. *Transportation Research Record*(1451).

Stephanie. (2018). What is a Stochastic Model? *Statistics How To*. 2018

Taha, H. A. (1996). *Riset Operasi* (Vol. 1). Jakarta: Binarupa Aksara.

Taha, H. A. (2004). *Operations research: An introduction (for VTU)*: Pearson Education India.

Widodo, S. (2013). Beberapa Permasalahan Dalam Penganggaran Modal. *MODUS, 25(2)*, 233-241.

Wijaya, A. (2012). Pengantar Riset Operasi. *Yogyakarta. Mitracanamedia.*

Wildan, W. R., Setyanto, N. W., & Rahman, A. (2014). Penjadwalan Pada Mesin Paralel Identik untuk Meminimasi Makespan dengan Menggunakan Pendekatan Mixed Integer Linear Programming (Studi Kasus pada PT Pertamina Production Unit Gresik–Pelumas). *Jurnal Rekayasa dan Manajemen Sistem Industri*, *2(5)*, p1112-1123.

Yamit, Z. (1993). *Manajemen Kuantitatif Untuk Bisnis (Operation Research)*: BPFE UGM.

YULIANTO, A.M. 2012. *Metode Transportasi Stepping Stone*. https://digensia.wordpress.com/2012/12/28/metode-transportasi-stepping-stone.

GLOSARIUM

Analisis Sensitifitas	: Suatu analisis untuk mengetahui akibat apa yang dihasilkan oleh perubahan parameter-parameter yang dianggap penting terhadap kinerja sistem. Analisa ini dilakukan degan tujuan mengantisipasi perubahan yang akan terjadi.
Asumsi	: Dugaan atau anggapan sementara yang belum dibuktikan benar atau tidak. Asumsi membutuhkan pembuktian langsung.
Batasan	: Adalah perhinggaan suatu pembahasan, batasan sangat penting dalam suatu model dengan tujuan mendapatkan hasil fungsi tujuan yang layak.
Buffer Stock	: Suatu tingkat persediaan tambahan yang disimpan untuk mengantisipasi resiko kekurangan persediaan. Kekurangan persediaan umumnya terjadi karena permintaan yang tidak pasti sehingga buffer stock sangat diperlukan untuk menangani hal tersebut.
Distribusi	: Suatu kegiatan yang bertujuan memasarkan barang dan jasa dari produsen kepada konsumen. Dalam distribusi parameter (barang apa, berapa banyak, berapa harganya, ditujukan kemana, kapan dibutuhkan) penting untuk ditentukan dengan tepat.
Free Software	: Software yang dapat diunduh dengan gratis oleh pengguna. Tidak jarang vendor dapat meminta biaya tambahan jika pengguna menginginkan custom software.
Ideal	: Ideal atau paling efektif dalam hal biaya adalah yang paling minimal sedangkan dalam hal keuntungan adalah yang paling maksimal.

Integer	: Tipe data yang merepresentasikan bilangan bulat bukan pecahan. Bilangan integer dapat bernilai positif maupun negatif.
Interface	: Dapat juga diartikan desain antarmuka penguna. Contohnya desain tampilan software, dan web yang ditujukan untuk pengguna. Interface berhubungan langsung dengan pengalaman penggun
Linier	: Persamaan aljabar dimana setiap sukunya mengandung konstanta, atau perkalian konstanta dengan variabel tunggal. Persamaan tersebut dikatakan linear sebab hubungan matematis ini dapat digambarkan sebagai garis lurus dalam Sistem koordinat Kartesius.
Matriks	: Sekumpulan bilangan yang disusun berdasarkan baris dan kolom. Banyaknya baris dan kolom pada suatu matriks ditunjukkan dalam ordo matriks tersebut.
Metode	: Cara yang dihunakan dalam menyelesaikan suatu permasalahan. Cara yang ditempuh tersebut dapat bermacam-macam namun tetap memiliki tujuan yang sama.
Model Konseptual	: Model konseptual umumnya digambarkan oleh suatu diagram dimana diagram tersebut dapat menjelaskan gambaran situasi pada sistem nyata yang diteliti.
Model Matematis	: Suatu pendekatan apa yang terjadi pada sistem nyata kedalam simbol-simbol matematis. Model matematis yang dibuat berhubungan dengan batasan dan asumsi yang ditentukan.
Optimal	: Suatu kondisi yang terbaik atau paling menguntungkan dalam suatu permasalahan.
Optimasi	: Proses pencarian hasil ideal (paling efektif) dari suatu permasalahan. Oprimasi dapat berupa memperbaiki sesuatu yang suah ada sebelumnya maupun membuat sesuatu yang baru sehingga memilihi hasil yang ideal.

Penugasan	: Pemberian tugas kepada sekelompok sumber daya yang akan melaksanakan tugas tersebut. Metode yang dibahas dalam penugasan bertujuan untuk menemukan solusi penugasan yang optimal.
Problem	: Suatu penyataan atas suatu keadaan yang tidak sesuai dengan harapan yang diinginkan.
R Package	: Serangkaian kode pemrograman yang dikembangkan oleh seseorang dengan tujuan menyelesaikan permasalahan tertentu.
Transhipment	: Suatu masalah transportasi yang membutuhkan kegiatan transit terlebih dahulu. Sebagai contoh barang yang dikirim dari manufaktur kepada konsumen. Dalam proses pengiriman tersebut barang tersebut harus disimpan dulu di suatu lokasi penyimpanan sementara sebelum akhirnya diteruskan ke konsumen.
Transportasi	: Perpindahan barang maupun manusia dari satu tempat ke tempat lainnya dengan menggunakan alat transportasi. Alat transportasi tersebut dapat digerakkan oleh mesin maupun manusia.
Variabel Keputusan	: Variabel yang dibuat untuk dapat menguraikan keputusan apa saja yang akan dibuat dalam masalah yang sedang diteliti.
Vektor	: Matriks yang hanya memiliki satu kolom atau satu baris saja. Disebut vektor kolom jika hanya memiliki sau kolom dan disebut vektor baris jika hanya memiliki satu baris.

INDEX

About the author: Ilyas Masudin

Ilyas Masudin is a lecturer and researcher in Industrial Engineering department at the University of Muhammadiyah Malang. His research interests are in the area of logistics and supply chain optimization. He received his bachelor's degree in Industrial Engineering from the University of Muhammadiyah Malang (2000). He holds a master's degree in Logistics & Supply Chain Management (MLSCM) from Curtin University of Technology (2007) and completed his PhD in Logistics from RMIT University (2012).

About the author: Muhammad Faisal Ibrahim

Muhammad Faisal Ibrahim, seorang dosen dan peneliti dari Program Studi Teknik Logistik Universitas Internasional Semen Indonesia dengan area penelitian Optimasi Sistem Rantai Pasok. Penulis lahir di Kota Samarinda Provinsi Kalimantan Timur pada tanggal 17 Desember 1993. Penulis menyelesaikan pendidikan jenjang Sarjana Strata 1 di Teknik Industri Universitas Muhammadiyah Malang pada tahun 2015. Setelah menyelesaikan pendidikan jenjang Sarjana penulis sempat bekerja sebagai praktisi selama kurang lebih enam bulan hingga akhirnya melanjutkan pendidikan jenjang Magister Strata 2. Penulis menyelesaikan pendidikan jenjang Magister Strata 2 di Program Pascasarjana Teknik Industri Institut Teknologi Sepuluh Nopember pada tahun 2017. Penulis dapat dihubungi melalui email faisalibrahim.ie@gmail.com.

Seiring berkembangnya bahasa pemrograman yang sangat cepat, menuntut kecepatan perkembangan keilmuwan lain yang menyertainya. Salah satu disiplin keilmuan teknik industri adalah optimasi. Disiplin ilmu ini erat kaitannya dengan pendekatan matematis yang mendekati sebuah permasalahan keteknikindustrian dari perspektif model matematis. Untuk mendapatkan hasil optimal dengan model matematis ini, salah satu metode yang paling populer dan banyak digunakan hampir di semua disiplin keilmuan di berbagai bidang adalah program linier (linear programming). Buku ini akan mencoba mengkombinasikan penyelesaian problem keteknikindustrian (problem solving) yang umum dijumpai di linear programming dengan pendekatan matematis dan heuristik dengan penyelesaian menggunakan bahasa pemrograman R.

ISBN 978-979-796-335-4

Kritik dan saran mengenai buku ini via email: ummpress@gmail.com

Penerbit Universitas Muhammadiyah Malang

Full Materi

ORIGINALITY REPORT

17%	16%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	faculty.nps.edu Internet Source	3%
2	docplayer.info Internet Source	1%
3	repository.binus.ac.id Internet Source	1%
4	www.omniascience.com Internet Source	1%
5	Submitted to University of Muhammadiyah Malang Student Paper	1%
6	arisbudi.staff.gunadarma.ac.id Internet Source	1%
7	Submitted to Oklahoma State University Student Paper	<1%
8	fr.scribd.com Internet Source	<1%
9	Submitted to Queen's University of Belfast Student Paper	<1%
	Submitted to University of Nottingham	

10	Student Paper	<1%
11	cran.r-project.org Internet Source	<1%
12	lp-solve.2324885.n4.nabble.com Internet Source	<1%
13	delfisuryani02.blogspot.com Internet Source	<1%
14	es.scribd.com Internet Source	<1%
15	repositori.uin-alauddin.ac.id Internet Source	<1%
16	www.arieanang.com Internet Source	<1%
17	pt.scribd.com Internet Source	<1%
18	Submitted to Harrisburg University of Science and Technology Student Paper	<1%
19	repository.usu.ac.id Internet Source	<1%
20	plato.la.asu.edu Internet Source	<1%
21	www.scribd.com Internet Source	<1%

22	www.gauss.pl Internet Source	<1%
23	edoc.site Internet Source	<1%
24	eprints.umm.ac.id Internet Source	<1%
25	docs.com Internet Source	<1%
26	datamining.dongguk.ac.kr Internet Source	<1%
27	www.coursehero.com Internet Source	<1%
28	library.binus.ac.id Internet Source	<1%
29	docobook.com Internet Source	<1%
30	epdf.tips Internet Source	<1%
31	eprints.uns.ac.id Internet Source	<1%
32	stiebanten.blogspot.com Internet Source	<1%
33	ssor.twi.tudelft.nl Internet Source	<1%
	es.slideshare.net	

34	Internet Source	<1%
35	vdocuments.site Internet Source	<1%
36	digilib.unila.ac.id Internet Source	<1%
37	www.tmi.sttal.ac.id Internet Source	<1%
38	Submitted to Universitas 17 Agustus 1945 Surabaya Student Paper	<1%
39	ml.scribd.com Internet Source	<1%
40	id.123dok.com Internet Source	<1%
41	yunisaindahcentyadewi.blogspot.com Internet Source	<1%
42	ukdw.ac.id Internet Source	<1%
43	myfatkhur.blogspot.com Internet Source	<1%
44	percolat.csa.iisc.ernet.in Internet Source	<1%
45	jrmsi.studentjournal.ub.ac.id Internet Source	<1%

46	cepeda-clasesdeinvope.blogspot.com Internet Source	<1%
47	adicelular.wordpress.com Internet Source	<1%
48	Ilyas Masudin, Tri Wastono, Fien Zulfikarijah. "The effect of managerial intention and initiative on green supply chain management adoption in Indonesian manufacturing performance", Cogent Business & Management, 2018 Publication	<1%
49	erwin2h.wordpress.com Internet Source	<1%
50	jurnal.tekmira.esdm.go.id Internet Source	<1%
51	d-nb.info Internet Source	<1%
52	cs.wmich.edu Internet Source	<1%
53	annawalyeni.blogspot.com Internet Source	<1%
54	link.springer.com Internet Source	<1%
55	Submitted to City University Student Paper	<1%
	www.databaser.net	

56	Internet Source	<1%
57	pt.slideshare.net Internet Source	<1%
58	bahasa-r.blogspot.com Internet Source	<1%
59	arifdhaniirwanto.blogspot.com Internet Source	<1%
60	eprints.soton.ac.uk Internet Source	<1%
61	comments.gmane.org Internet Source	<1%
62	Submitted to CSU, Long Beach Student Paper	<1%
63	anggarakidal12.blogspot.com Internet Source	<1%
64	gatiagusti.blogspot.com Internet Source	<1%
65	Christian H.C.A. Henning, Arne Henningsen. "Modeling Farm Households' Price Responses in the Presence of Transaction Costs and Heterogeneity in Labor Markets", American Journal of Agricultural Economics, 2007 Publication	<1%
66	jibrael-rasta.blogspot.com	

	Internet Source	<1%
67	R. B. Mallick. "Development and evaluation of a field permeameter as a longitudinal joint quality indicator", International Journal of Pavement Engineering, 3/1/2006 Publication	<1%
68	Malak , Malakeh | Al Maharmeh , Ahmad. "Socialization Into Nursing", Publication	<1%
69	عاشور ، محمد السيد. "الوحدة الإسلامية في إطار العلاقات الدولية و التغيرات العصرية : دراسة مقارنة", Publication	<1%
70	text-id.123dok.com Internet Source	<1%
71	blog.gmane.org Internet Source	<1%
72	www.competitive.org.ph Internet Source	<1%
73	Submitted to Isra University Student Paper	<1%
74	mylaporanti.blogspot.com Internet Source	<1%
75	mochamadfamas.blogspot.com Internet Source	<1%
76	www.spotseven.de Internet Source	<1%

77	karlarao.github.io Internet Source	<1%
78	muhammadisramaulana.blogspot.com Internet Source	<1%
79	eprints.uny.ac.id Internet Source	<1%
80	media.neliti.com Internet Source	<1%
81	misranindustri.blogspot.com Internet Source	<1%
82	repository.widyatama.ac.id Internet Source	<1%
83	ejournal.unsrat.ac.id Internet Source	<1%
84	centerscm.org Internet Source	<1%
85	hanifmisfir025.blogspot.com Internet Source	<1%
86	www.analytics-tuts.com Internet Source	<1%
87	github.molgen.mpg.de Internet Source	<1%
88	Submitted to University of Edinburgh Student Paper	<1%

89	www.medepi.net Internet Source	<1%
90	www.gudanglinux.info Internet Source	<1%
91	penguinmerah.org Internet Source	<1%
92	www.researchgate.net Internet Source	<1%
93	repository.uinjkt.ac.id Internet Source	<1%
94	journal.pim.ac.th Internet Source	<1%
95	Ganesh Subramaniam. "Simulation-based optimisation for material dispatching in Vendor-Managed Inventory systems", International Journal of Simulation and Process Modelling, 2007 Publication	<1%
96	www.seputartradingforex.com Internet Source	<1%
97	dl4a.org Internet Source	<1%
98	managementaccess.blogspot.com Internet Source	<1%
99	www.netsuccess.gr Internet Source	<1%

100	www.macmillanhighered.com Internet Source	<1%
101	widuri.raharja.info Internet Source	<1%
102	tvschool.alazhar-cibubur.sch.id Internet Source	<1%
103	www.promtools.org Internet Source	<1%
104	kaskus-ind.blogspot.com Internet Source	<1%
105	www.ea.govt.nz Internet Source	<1%
106	zuuhandri.blogspot.com Internet Source	<1%
107	zh.scribd.com Internet Source	<1%
108	www.cyberdriveillinois.com Internet Source	<1%
109	www.zib.de Internet Source	<1%
110	Siriphong Lawphongpanich. "Dynamic Slope Scaling Procedure and Lagrangian Relaxation with Subproblem Approximation", Journal of Global Optimization, 05/2006 Publication	<1%

111	read.cucdc.com Internet Source	<1%
112	anzdoc.com Internet Source	<1%
113	rangga-shavin.blogspot.com Internet Source	<1%
114	id.scribd.com Internet Source	<1%
115	blog.unnes.ac.id Internet Source	<1%
116	rusdiantidian7.blogspot.com Internet Source	<1%
117	Pierre Lafaye de Micheaux, Rémy Drouilhet, Benoit Liquet. "The R Software", Springer Nature America, Inc, 2013 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		